NASA Technical Memorandum 108791

User’s Manual for the HYPGEN Hyperbolic Grid Generator and the HGUI Graphical User Interface

William M. Chan, Ing-Tsau Chiu, and Pieter G. Buning

October 1993

National Aeronautics and
Space Administration

Table of Contents

1. Introduction and Overview

This manual is a combined documentation for version 2.0 of the *HYPGEN* hyperbolic grid generator and the *HGUI* graphical user interface. The *HYPGEN* program is used to generate a 3-D volume grid over a user-supplied single-block surface grid utilizing the 3-D hyperbolic grid generation equations [1]. It is a fully self-contained code and can be run in batch mode on any machine supporting Fortran 77. Users on Silicon Graphics (SGI) workstations have the option to use the graphical user interface *HGUI* for preparing and fine tuning the input parameters. Hooks to *HYPGEN* and *PLOT3D* [2] are provided for grid generation and visualization, respectively.

Overviews of the flow of the input and output files associated with running *HYPGEN* and *HGUI* are illustrated in Figures 1.1 and 1.2. More details of these files are given in §2.2. The double arrows for the input parameters file in Figure 1.2 indicate that if this file exists, then *HGUI* would read it in as input. In any case, the parameters in the file can be initialized and modified via *HGUI* .

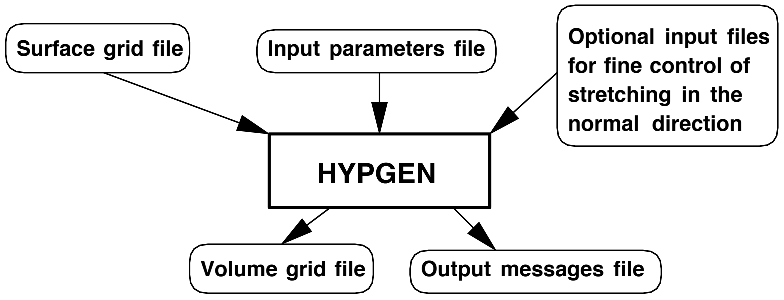


Figure 1.1. Overview of input and output files for running *HYPGEN* in batch mode.

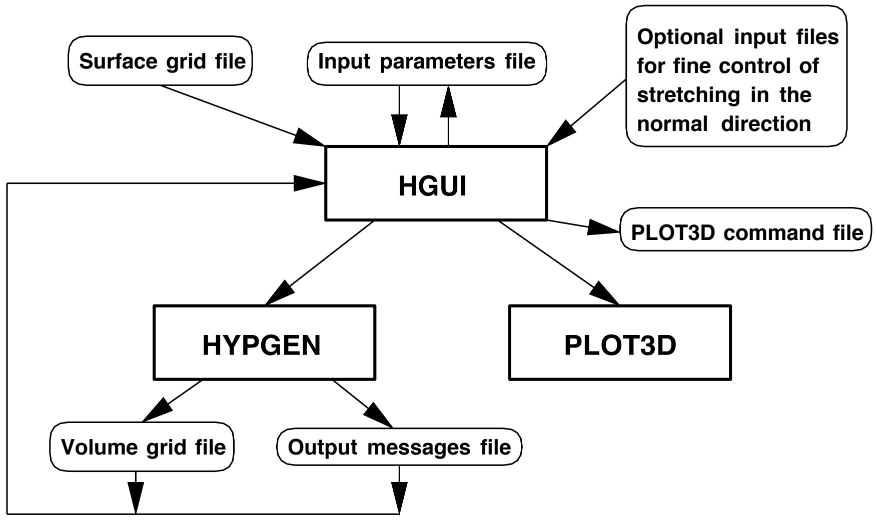


Figure 1.2. Overview of *HGUI* with its hooks and input and output files.

Section 3 of this manual may be skipped by users running *HYPGEN* in stand-alone mode without *HGUI* .

The filenames of the various input and output files used for running *HYPGEN* in batch mode are summarized in Table 1.1. These files are described in detail in §2.2.

	Filename	Description	req./opt.
Input	*surf.i*	surface grid	required
Input	< *standard input*	input parameters	required
Input	*zetavar.i*	variable far field, initial/end spacing	optional
Input	*zetastr.i*	user-defined stretching function	optional
Output	*plot3d.dat*	volume grid	
Output	> *standard output*	output messages	

Table 1.1. Input and output filenames for running *HYPGEN* in batch mode.

The filenames of the various input and output files used for running *HYPGEN* using *HGUI* are summarized in Table 1.2. The contents of these files are identical to those in Table 1.1. *HGUI* uses a *basename* convention where the *basename* is typically the name of the grid and extensions are added to the *basename* to identify the different input and output files. More details are given in §3.2.1 on how to set the *basename*.

	Filename	Description	req./opt.
Input	*basename2d.dat*	surface grid	required
Input	*basename.i*	input parameters	optional
Input	*basenamevar.dat*	variable far field, initial/end spacing	optional
Input	*basenamestr.dat*	user-defined stretching function	optional
Output	*basename3d.dat*	volume grid	
Output	*basename.out*	output messages	
Output	*basename.i*	modified input parameters	
Output	*basename.com*	*PLOT3D* commands	

Table 1.2. Input and output filenames for running *HYPGEN* using *HGUI* .

2. HYPGEN Hyperbolic Grid Generator

2.1. Introduction to the *HYPGEN* Program

The *HYPGEN* program is used to generate a 3-D volume grid over a user-supplied single-block surface grid by marching away from the initial surface with the step size given by a stretching function in the normal direction. This is accomplished by solving the 3-D hyperbolic grid generation equations (two orthogonality relations and one cell volume constraint) [1]. A 2-D grid can also be generated by specifying appropriate boundary conditions.

The program is evolved from a 3-D hyperbolic grid generation program *(HYG3D)* by Steger and Rizk [1]. In previous versions of the program, special care and parameter adjustments were frequently necessary in order to cope with a new geometry that was significantly different from ones treated before. The current *HYPGEN* program is much more modular; many new features have been added and many new techniques have been implemented to improve the robustness [3]. The ultimate goal in future versions of the program is for it to be able to automatically select as many parameters as possible such that the amount of user input is reduced to a minimum.

The notation used in the manual is for a 3-D grid in generalized coordinates ξ, η, ζ which correspond to the J, K and L directions, respectively. The user-supplied surface grid is defined in the J and K directions while the L-direction is the normal or marching direction.

2.1.1. Overview of the Grid Generation Procedure

The following is a step by step guideline for generating a volume grid.

(1) Prepare input surface grid file (see §2.2.1, 2.3.5).
(2) Prepare input parameters file (see §2.2.1, 2.3.1, 2.3.2, 2.3.3, 2.3.4). Users on Silicon Graphics workstations have the option to employ the graphical user interface *HGUI* (see §3).
(3) Compile source code (see §2.5.).
(4) Run program interactively (see §2.5.).
(5) View the grid and adjust input parameters if necessary (see §2.3.1, 2.3.4, 2.4.1, 2.4.2).

2.2. Input and Output Files

The input and output files used by *HYPGEN* are described in this section. The filenames assumed for these files are given in the following subsections.

2.2.1. Summary of Input Files

The input files required from the user are: (i) the surface grid file and (ii) the input parameters file. Certain parameter options for the stretching in the normal direction also require one of the following files: (iii) the variable far field and initial/end grid spacings file and (iv) the user-defined stretching function file (see §2.3.1, 2.3.2). Automatic checks are performed by the code to determine whether the files (i), (iii) and (iv) are formatted or unformatted. These input files are described in more detail below where the read formats are illustrated using unformatted read statements.

(i) **Surface grid file** (required)

The surface grid (L = 1) with dimensions $(jmax, kmax, 1)$ in *PLOT3D* [2] single block format has to be supplied in a file named *surf.i*. The read format is

$$\begin{aligned}&\text{read(unit) } jmax, kmax, ldum\\ &\text{read(unit) } ((x(j,k), j = 1, jmax), k = 1, kmax),\\ &\qquad\qquad ((y(j,k), j = 1, jmax), k = 1, kmax),\\ &\qquad\qquad ((z(j,k), j = 1, jmax), k = 1, kmax)\end{aligned}$$

If the grid is not periodic in the J-direction, then $jmax$ is the number of grid points in the J-direction. If the grid is periodic in the J-direction, then the J $= jmax$ point is assumed to be coincident with the J = 1 point. The same applies for the K-direction. Note that $ldum$ should be equal to one. By setting $jmax = 1$ or $kmax = 1$, a 2-D grid in the K-L or J-L plane are obtained respectively.

(ii) **Input parameters file** (required)

The input parameters file is read from standard input (see §2.5). The user is free to choose any name for this file. An alternative to supplying this file is to use the graphical user interface *HGUI* to initialize the input parameters. A summary and guidelines on the choice of parameters are given in §2.3.

(iii) **Variable far field, initial/end grid spacing file** (optional)

For each point on the surface grid, a different far field distance, initial and/or end grid spacing in the marching direction can be specified (see §2.3.2 for more details). This information is read in from the file *zetavar.i* as follows:

$$\begin{aligned}&\text{read(unit) } jmax, kmax, ldum\\ &\text{read(unit) } (((zetavar(j,k,n), j = 1, jmax), k = 1, kmax), n = 1, 3)\end{aligned}$$

where $n = 1$ corresponds to the far field distance, $n = 2$ corresponds to the initial spacing and $n = 3$ corresponds to the end spacing in the marching direction. Note that this file has the same dimensions as the surface grid file *surf.i* (with $ldum = 1$ assumed).

(iv) **User-defined stretching function file** (optional)

A user-defined stretching function in the normal direction can be read in from the file *zetastr.i* which has the format

$$\begin{aligned}&\text{read(unit) } jzs, kzs, lmax\\ &\text{read(unit) } (((zetastr(j,k,l), j = 1, jzs), k = 1, kzs), l = 1, lmax)\end{aligned}$$

where $jzs = 1$ or $jmax$ and $kzs = 1$ or $kmax$, and $lmax$ is the number of points in the normal direction (see §2.3.2 for more details).

2.2.2. Summary of Output Files

Two output files are produced by *HYPGEN* : (i) the 3-D volume grid file and (ii) the output messages file. These are described in more detail below.

(i) **Volume grid file**

The output 3-D volume grid file can be either unformatted (specified by the input parameter IFORM = 0) or formatted (IFORM=1) and it is written to a file named *plot3d.dat* which is in *PLOT3D whole* format:

$$\begin{aligned}&\text{write(unit) } jmax, kmax, lmax\\ &\text{write(unit) } (((x(j,k,l), j=1,jmax), k=1,kmax), l=1,lmax),\\ &\qquad\qquad (((y(j,k,l), j=1,jmax), k=1,kmax), l=1,lmax),\\ &\qquad\qquad (((z(j,k,l), j=1,jmax), k=1,kmax), l=1,lmax)\end{aligned}$$

(ii) **Output messages file**

The output file directed to standard output contains the input parameters, warning and error messages, and the output from the cell volume and Jacobian checks. The user is free to pick any name for this file (see §2.5).

2.3. Input Parameters and Surface Grid

The options available for each input parameter are summarized in §2.3.1 while more detailed explanations of these parameters are given in the subsequent subsections. Details of the different methods to specify the stretching function that govern the grid point distribution in the normal direction are given in §2.3.2. The various boundary condition types are explained in §2.3.3. Guidelines on the choice of the input parameters related to grid smoothing are discussed in §2.3.4. Some basic rules on the construction of a good surface grid are given in §2.3.5.

2.3.1. Summary of Input Parameters

A typical input parameters file contains the following lines where the text at the end of each line is for comment purpose only and can be omitted.

```
0                     IFORM(0/1)
1,1                   IZSTRT(-1/1/2),NZREG
41,0.9,1.0e-5,0.0     NPZREG(),ZREG(),DZ0(),DZ1()
1,1,3,7               IBCJA,IBCJB,IBCKA,IBCKB
1,0.01,15             IVSPEC(1/2),EPSSS,ITSVOL
2,0.6                 IMETH(0/1/2/3),SMU2
0.0,0.0               TIMJ,TIMK
```

The various options available for the parameters in the input parameters file are summarized in this section.

IFORM	= 0 for unformatted output of *PLOT3D* volume grid file
	= 1 for formatted output of *PLOT3D* volume grid file
IZSTRT	= 1 for exponential stretching in L
	= 2 for hyperbolic tangent stretching in L
	= −1 for user-defined stretching read in from *zetastr.i* (†)

NZREG = number of L-regions

† NPZREG, ZREG, DZ0, DZ1 are disregarded; NZREG= 1 is assumed

Repeat NPZREG, ZREG, DZ0, DZ1 for each L-region
NPZREG = number of points (including ends) in the L-region

ZREG > 0 approximate distance to march out in the L-region
≤ 0 variable far field distance read in from *zetavar.i* (*)

DZ0 > 0 initial spacing in the L-region
$= 0$ initial spacing not fixed in the L-region
< 0 variable initial spacing read in from *zetavar.i* (*)

DZ1 > 0 end spacing in the L-region
$= 0$ end spacing not fixed in the L-region
< 0 variable end spacing read in from *zetavar.i* (*)

* variations are assumed in the first L-region if more than one L-region is specified

The boundary condition types at J = 1, J = *jmax*, K = 1, K = *kmax* are indicated by IBCJA, IBCJB, IBCKA, IBCKB, respectively

IBCxx = -1 float x, y and z (free floating)
= -2 to -1000 for outward-splaying free floating boundary condition condition which bends the edge away from the interior. Use small $|IBC|$ for small bending.
= 1 fix x, float y and z (constant x plane)
= 2 fix y, float x and z (constant y plane)
= 3 fix z, float x and y (constant z plane)
= 4 float x, fix y and z
= 5 float y, fix x and z
= 6 float z, fix x and y
= 7 floating collapsed edge with matching upper and lower sides
= 10 periodic condition in J (**)
= 11 reflected symmetry condition with $x = 0$ plane
= 12 reflected symmetry condition with $y = 0$ plane
= 13 reflected symmetry condition with $z = 0$ plane
= 20 singular axis point (‡)
= 21 constant x planes for interior and boundaries slices (**)
= 22 constant y planes for interior and boundaries slices (**)
= 23 constant z planes for interior and boundaries slices (**)

xx represents JA, JB, KA or KB.

** must also apply for the other end condition in the same direction, i.e., IBCJA = IBCJB or IBCKA = IBCKB.

‡ This condition can be used for IBCJA, IBCJB only. If a singular axis point occurs at a K boundary, the user has to interchange the J and K families for the grid generation process.

Default values for the following grid smoothing parameters are given in [ ] brackets.

IVSPEC = 1 control volumes computed by area element times arc length [1]
= 2 control volumes computed by scaling with volume of a sphere (closed body scaling assumed when both $j = 1$ and *jmax* are singular axis points; otherwise, open body is assumed)

EPSSS = parameter governing how fast the volumes become those of a sphere O(0.001-0.05) (activated only if IVSPEC = 2) [0.01]

ITSVOL = number of times specified volumes are smoothed [5]

IMETH	$= 0$ for constant smoothing coef.
	$= 1$ for spatially varying smoothing coef.
	$= 2$ for spatially varying smoothing coef. with implicit averaging [2]
	$= 3$ for spatially varying smoothing coef. with angle-bisecting predictor
SMU2	= 2nd order explicit smoothing coef., $O(1)$ [0.5]
TIMJ	= Barth implicitness factor in J - set up to 3.0 for large concave geometry in the J-direction, set to 0.0 otherwise [0.0]
TIMK	= Barth implicitness factor in K - set up to 3.0 for large concave geometry in the K-direction, set to 0.0 otherwise [0.0]

If IBCJA $\neq 20$ and IBCJB $\neq 20$, IAXIS, EXAXIS and VOLRES need not be specified.

IAXIS	$= 1$ for mixed extrapolation with volume scaling [1]
	$= 2$ for local central differencing
EXAXIS	= order of extrapolation at axis which controls pointedness (0.0-1.0) [0.3]
VOLRES	= scale factor for volumes at one point from axis. This is activated only when EXAXIS is non-zero O(0.1-0.8) [0.4]

Users with input parameters file for version 1.3 or earlier of *HYPGEN* can use the tool *CONVI* (Fortran source code *convi.f*) to convert to the new format for the input parameters file. This can be done by executing *CONVI* as follows:

`convi` < *old_input_parameters_filename* > *new_input_parameters_filename*

2.3.2. Stretching Function Control in the Normal Direction

HYPGEN offers a variety of ways to specify the stretching function in the normal direction. These are described in detail below.

(a) *Exponential or hyperbolic tangent stretching with constant initial/end grid spacing and far field distance*

Two types of built-in stretching functions are available: the exponential stretching (IZSTRT= 1) and the hyperbolic tangent stretching [4] (IZSTRT= 2). For the exponential stretching (also known as the geometric stretching), the user specifies the initial spacing, the distance to march out and the number of points to be used (NPZREG). A grid spacing stretching ratio (ratio of grid spacing at level L relative to that at level L -1) is then determined and is maintained in the entire domain. For the hyperbolic tangent stretching, in addition to specifying the distance to march out and the number of points to be used, the grid spacing at one or both ends of the domain may be specified. The hyperbolic tangent stretching should be used with caution since the grid spacings specified by the user at the ends of the domain are only the asymptotic values, i.e., the actual grid spacings computed for the 1-D stretching are usually slightly different from the ones specified, depending on the domain size and the number of points used. If only the initial spacing is specified, the hyperbolic tangent stretching tends to place slightly more points in the far field region than the exponential stretching.

The stretching in the normal direction can be split into L-regions where each region can have a different size, number of points and grid spacing specifications (see the tank example in §2.6). If the initial spacing for an outer L-region is set to zero in the input file, then the initial spacing is computed by maintaining the same stretching ratio at the last grid point from the previous inner region, thus providing a smooth transition between regions.

(b) *Exponential or hyperbolic tangent stretching with variable initial/end grid spacing and far field distance*

In certain applications, it is desirable to have different far field distances at different locations, e.g., a far field shell that closely follows a bow shock (a small far field distance near the nose of the vehicle and a large far field distance downstream). Similarly, a variable initial grid spacing may be desirable along the wake cut of an airfoil C-mesh to fan out the points at the downstream boundary. For each point on the surface grid, *HYPGEN* allows the user to specify the following three quantities: a far field distance (ZREG), an initial and an end grid spacing (DZ0 and DZ1). If any of ZREG, DZ0 or DZ1 is negative in the input parameters file, the file *zetavar.i* is read in (see §2.2.1 for format). A negative value for any one of ZREG, DZ0 or DZ1 implies that input will be taken from *zetavar.i* while a positive value implies that input will be taken from the constant value in the input parameters file. If more than one L-region is specified in the input, the values from *zetavar.i* are assumed to apply in the first L-region, while the constant values specified in the input parameters file are assumed in subsequent L-regions.

An interactive tool *SETZETA* (Fortran source code *setzeta.f*) is available to assist the user in initializing the file *zetavar.i*. For each of the three fields for ZREG, DZ0 and DZ1, the user can choose a constant distance/spacing or a piecewise linear variation of distance/spacing in both the J and/or K directions. Node points and distances/spacings at the node points can be prescribed and a linear interpolation based on the physical distance between node points is performed. For the far field distance, an extra option is available to compute the distance from each point on the surface to the centroid at each J or K constant station. This is useful for internal grid generation (see duct example in §2.6). Work is in progress to develop more options to be included in the future.

This tool is not intended to cover all possible ways to specify the three fields but rather to serve as a means to quickly initialize the fields for certain simple cases. For more complex cases, the user can write a separate program (or add a special subroutine to *setzeta.f*) to generate the file *zetavar.i* that is tailored to his/her specific requirements.

Input parameters for *SETZETA* are prompted from the user interactively. If the file *setzeta.i* does not exist, the input parameters from the user are written out to a new file *setzeta.i*. This allows the user to edit the input parameters and also allows the code to be run again by taking input from *setzeta.i*.

(c) *User-defined stretching*

There are certain situations where a special stretching function, other than exponential or hyperbolic tangent, is required. By specifying IZSTRT $= -1$, a user-defined stretching function is read in from the file *zetastr.i* with dimensions $jzs \times kzs \times lmax$. Four options are provided depending on the values of jzs, kzs:

$jzs = 1,$	$kzs = 1$	- same stretching for all J and K
$jzs = jmax,$	$kzs = 1$	- copy variation in J for K = 1 to all K = const. slices
$jzs = 1,$	$kzs = kmax$	- copy variation in K for J = 1 to all J = const. slices
$jzs = jmax,$	$kzs = kmax$	- different stretching for each J and K

A stretching function for point (J,K) is defined by a monotonically increasing one-dimensional array of numbers with index running from 1 to $lmax$. The approximate distance of the Lth layer of the volume grid from the body surface is specified by the Lth element of the array; i.e., the first element has the value zero, the last element is assigned the far field distance, and the grid spacing between consecutive L-shells is given by the difference between the corresponding consecutive elements in the array. If a user-defined stretching function is used, then the variable far field distance and initial/end grid spacings option cannot be activated, and vice versa.

2.3.3. Explanation of Boundary Condition Types

The various boundary condition types in §2.3.1. are described in detail below.

Free floating/splay option (IBC < 0)

The free floating boundary condition (IBC $= -1$) is used when there is no particular restriction needed at the boundary. When it is desirable to make the grid bend away from the interior, the splay option (IBC < -1) can be used (see Figure 2.1a,b).

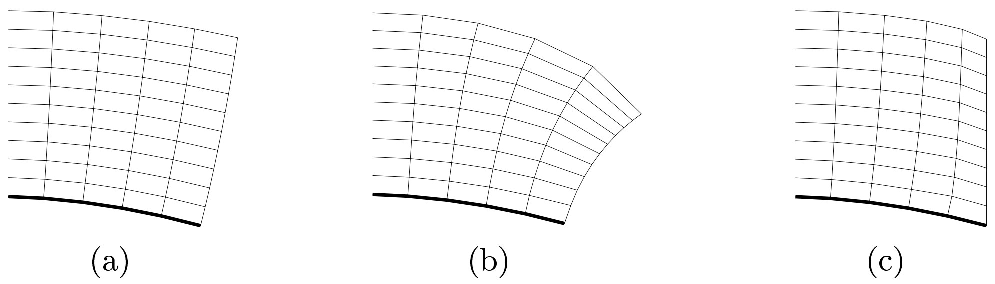


Figure 2.1. A slice through a volume grid with (a) a free floating boundary, (b) a free floating boundary with splay, and (c) a constant plane boundary (━ surface grid line, — volume grid line).

Constant x, y or z planes (IBC $= 1, 2, 3$)

This boundary condition type is used to ensure that the volume grid maintains a constant x, y or z plane at a particular boundary (see Figure 2.1c). Note that this can only hold if the boundary on the supplied surface grid lies on the specified constant plane. If the boundary of the surface grid does not lie on a constant plane, then the specified coordinate will be fixed locally at each point on the boundary.

The specified constant plane is allowed to be non-orthogonal to the surface grid including the case where it is tangential to the surface grid. *HYPGEN* automatically relaxes the orthogonality constraint for the points near the boundary so that a smooth transition is achieved from the non-orthogonal boundary points to the nearly orthogonal interior points.

Float one coordinate, fix the other two (IBC = 4, 5, 6)

This boundary condition type is used to ensure that the volume grid grows along the x (IBC = 4), y (IBC = 5) or z (IBC = 6) direction at a particular boundary. For example, figure 2.2 shows various slices of the volume grid for half of a frustum where the top boundary is made to grow along the z direction (IBC = 6) while the bottom boundary is made to maintain a constant z plane (IBC = 3).

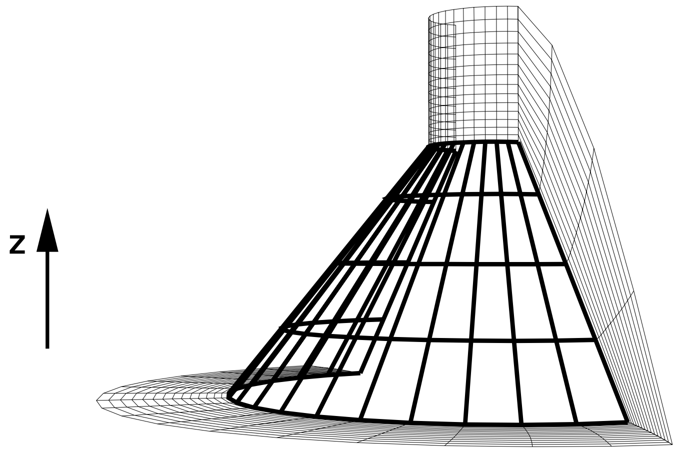


Figure 2.2. Various computational planes of the volume grid for half of a frustum (━━ surface grid line, —— volume grid line).

Floating collapsed edge with matching upper and lower sides (IBC = 7)

This boundary condition is applied at a boundary consisting of an upper and lower edge that are collapsed to a slit. Each point on the upper edge is assumed to match with a point on the lower edge except at the leading and trailing edge. The collapsed edge can be of either a C- or O-type topology (see the wing examples in §2.6 for illustrations). For example, if the K = *kmax* edge is a collapsed edge, then K = constant grid lines (variations in J) can be of the C-type (non-periodic) or of the O-type (periodic) and *jmax* has to be odd. For the O-type grid, the *jmax* point is assumed to be coincident with the J = 1 point which is assumed to be at either the leading or trailing edge. Grid points on the upper and lower sides of the collapsed edge are obtained by a floating condition from each side followed by averaging which results in a coincident point for both upper and lower sides.

Periodic (IBC = 10)

This boundary condition is used for grids with a periodic direction. The first and last grid point in the periodic direction of the supplied surface grid are assumed to be coincident.

Reflected symmetry (IBC = 11, 12, 13)

This boundary condition type assumes that the grid point *next* to the boundary lies on the $x = 0$ or $y = 0$ or $z = 0$ plane (symmetry plane), and that the grid point on the boundary is the reflection of the second grid point from the boundary about the symmetry plane (see Figure 2.3b). The reflected point on the other side of the symmetry plane is required by certain flow solvers.

Singular axis point (IBC $= 20$)

This boundary condition is available only in the J direction. If a singular axis point condition is applied at J $= 1$, then it is assumed that all the points at J $= 1$ and all K have the same exact coordinates on the surface grid (L $= 1$) and that this is maintained at each subsequent L-shell; i.e, points on J $= 1$ for all K and L form a singular axis. Also it is assumed that the boundary condition for the K boundaries is either periodic, reflected symmetry or constant plane (see Figure 2.3a-c). A distinction is made for the case where J $= 1$ is a singular axis point on the surface but it is desired that the points for different K fan out in the L-direction. For this case, a floating condition (IBCJA $= -1$) may be applied.

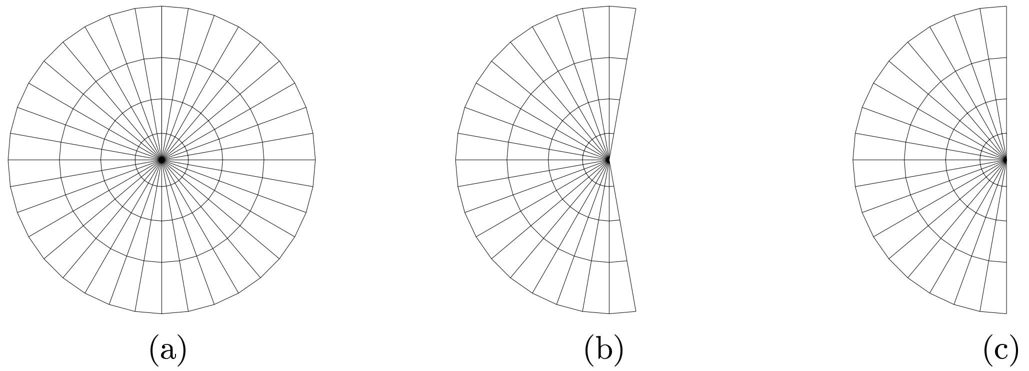


Figure 2.3. A surface grid where J $= 1$ is a singular axis point and K is the circumferential direction with a (a) periodic condition, (b) reflected symmetry condition, (c) constant plane condition at the K $= 1$ and *kmax* boundaries.

Constant interior and boundary planes (IBC $= 21, 22, 23$)

This boundary condition type is used to ensure that a constant x, y or z plane is maintained for all planes in J or K at both the interior and boundary. For example, certain space marching schemes require that each streamwise slice of the volume grid lie on a constant plane.

2-D grids

Two-dimensional grids can be generated by *HYPGEN* as follows. If the mesh is 2-D in K, then the supplied surface mesh should have $kmax = 1$. The boundary conditions should be set with IBCKA $=$ IBCKB $= 1$, 2 or 3 depending on the coordinate that is held constant in the 2-D plane. Similar conditions hold for a 2-D mesh in J.

2.3.4. Guidelines on the Choice of Grid Smoothing Parameters

The input parameters can be classified to fall in one of the following four categories:

(1) Output format control: IFORM

(2) Stretching function control in the normal direction: IZSTRT, NZREG, NPZREG, ZREG, DZ0, DZ1

(3) Boundary conditions: IBCJA, IBCJB, IBCKA, IBCKB

(4) Grid smoothness and quality controls: IVSPEC, EPSSS, ITSVOL, IMETH, SMU2, TIMJ, TIMK, IAXIS, EXAXIS, VOLRES

The parameters in categories 1, 2 and 3 have already been explained in the previous subsections. Grid smoothness control is governed by the parameters in category 4 and the following notes serve to help in their selection. An example is given at the end of this subsection to illustrate the effects of varying the three most important of these parameters.

(a) Control volumes specification (IVSPEC, EPSSS, ITSVOL) -

The type of volume specification is given by IVSPEC. Selection of IVSPEC $= 1$ (cell area times the normal marching distance) should produce satisfactory results for most cases. In some situations, using IVSPEC $= 2$ can bring in extra smoothing to the grid by scaling with the volumes of a sphere. With this option, the longest dimension of the body is assumed to be along the J-direction. The body is assumed closed when singular axis conditions are applied at both ends in J; otherwise it is assumed opened. EPSSS is only used with IVSPEC $= 2$ and it controls how rapidly the grid volumes are blended to become the grid volumes for a sphere as the grid marches out in L. Increasing EPSSS exponentially increases the spherical volume blending. This option tends to make the cell volumes more uniform in the far field and tends to make the shape of the far field boundary approach that of a sphere.

Increasing ITSVOL has the effect of increasing the averaging of volumes between grid cells. This also has a smoothing effect on the grid but is not as strong as that provided by SMU2. A small or zero ITSVOL (< 5), together with a small SMU2 ($<\approx 0.5$), should be used with IVSPEC $= 1$ if it is desired to have the grid points maintain a similar distribution at each L-shell as that on the surface grid. A higher ITSVOL will tend to spread out the grid points more. Very high values of ITSVOL ($>\approx 100$) may help to spread out coverging grid lines in tight concave corners but may have the undesirable effect of over-distorting the cell volumes at other places.

(b) Methods and smoothing coefficients (IMETH, SMU2) -

A spatially variable smoothing coefficient (IMETH ≥ 1) is recommended for most cases. With IMETH ≥ 1, the level of smoothing at each point in the grid is automatically adjusted [3]. The user only needs to enter the scale factor SMU2. The algorithm ensures that the smoothing is small near the body surface to preserve orthogonality. Smoothing is increased locally by detecting whether grid lines are converging as in concave regions. When very sharp convex corners are present in the geometry, IMETH $= 2$ or 3 is recommended. It is found from experience that IMETH $= 2$ is more suitable for some geometries while IMETH $= 3$ is better for others. The grid smoothness at a convex corner is enhanced if SMU2 and ITSVOL are not too large when IMETH$= 2$ or 3 is activated. This is not always possible when concave corners are also present. Thus several parameter adjustments may be needed to arrive at an acceptable result. In some situations, a high smoothing is needed near the body surface. This is provided by setting a constant smoothing coefficient with IMETH = 0.

The scale factor for the smoothing coefficient is given by SMU2 which is typically $O(1)$. For cases with extreme concave topologies, SMU2 can be $O(10)$ with IMETH ≥ 1. The strategy for selecting the "best" value of SMU2 is that it should be as small as possible to maintain orthogonality but large enough to provide sufficient smoothness to the grid.

(c) TIM factors (TIMJ, TIMK) -

TIMJ is an implicitness factor introduced by Barth [5] to help spread out grid lines that might converge as the grid is marched out. If there are no large concave corners in the ξ direction, TIMJ should be set to zero. Otherwise, TIMJ can be increased to a maximum of about 4.0 for very severe concave corners in ξ. TIMK is prescribed similarly in the η direction.

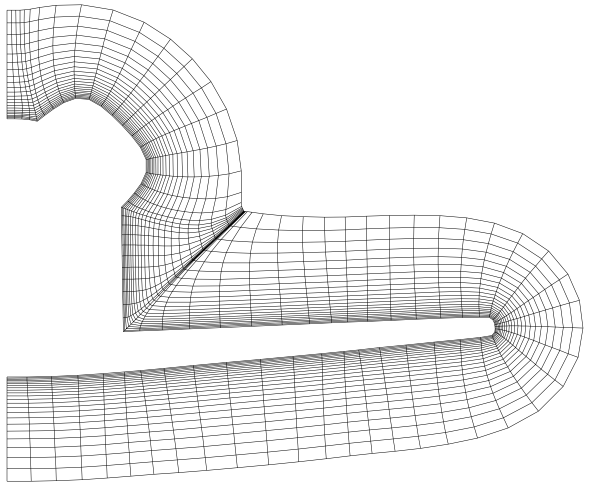
(d) Axis parameters (IAXIS, EXAXIS, VOLRES) -

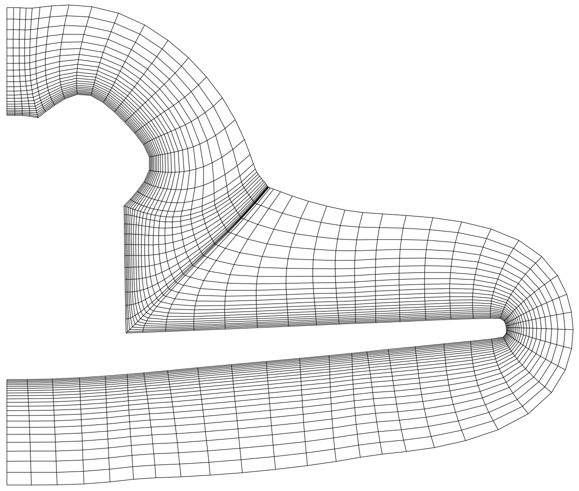
Since IAXIS = 2 is still in the experimental stage, IAXIS = 1 is recommended and should work for most cases. The pointedness of the grid at the axis is controlled by EXAXIS. This parameter should be in the range 0.0 to 1.0. If the axis is too blunt, EXAXIS should be increased and if it is too pointed, it should be decreased. VOLRES controls the volume scaling near the axis. It is activated only when EXAXIS is non-zero and should be used with ITSVOL ≈ 3 or more. If the volumes near the axis are too large, VOLRES should be reduced and vice versa.

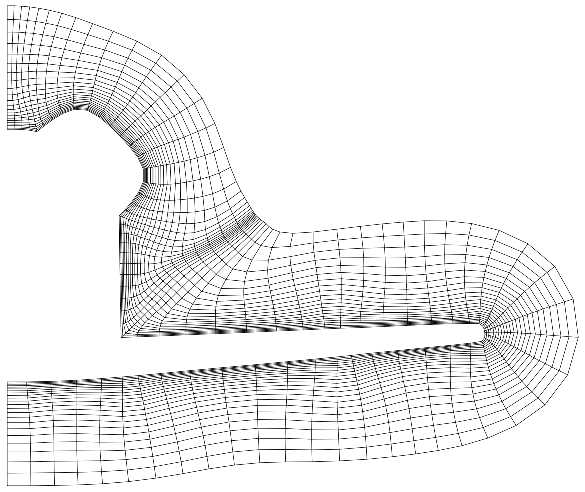
Example

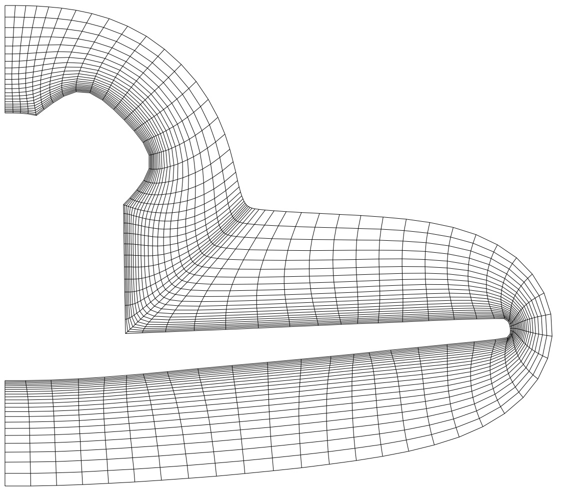
For most cases, a proper choice of TIMJ/K, ITSVOL and SMU2 is sufficient to produce a good quality grid. The effects of varying these parameters are illustrated in Figure 2.4a-e. Figure 2.4a is a base case where TIMJ and ITSVOL are set to zero and a small SMU2 is used. Orthogonality of grid lines is well maintained but the grid lines tend to bunch together in the concave corner. By increasing TIMJ, the grid lines are made to spread out more but not enough to produce a satisfactory grid (Figure 2.4b). Increasing ITSVOL alone has the effect of averaging the local cell volumes with their neighbors but the smoothness of the grid lines has been degraded in the concave corner (Figure 2.4c). By increasing SMU2 alone, the grid lines become very smooth and spread out in the concave corner, but orthogonality is lost in other areas (Figure 2.4d). Note that the grid is also pulled in more in convex regions (see the tip region in the far right of Figure 2.4d). Figure 2.5e shows the effect of a combination of the three parameters. An intermediate value of SMU2 is used to balance smoothness with orthogonality. TIMJ and ITSVOL are chosen to enhance spreading of grid lines in the concave region.

It is difficult to describe the effects of all possible combinations of parameters for a variety of geometric features. A flavor of the effects of the three most important parameters is given above. Since the execution speed of the code is so fast, the user can accumulate experience quickly by trying different combinations of parameters.


(a) TIMJ=0, ITSVOL=0, SMU2=0.5


(b) TIMJ=3, ITSVOL=0, SMU2=0.5


(c) TIMJ=0, ITSVOL=100, SMU2=0.5


(d) TIMJ=0, ITSVOL=0, SMU2=2.5

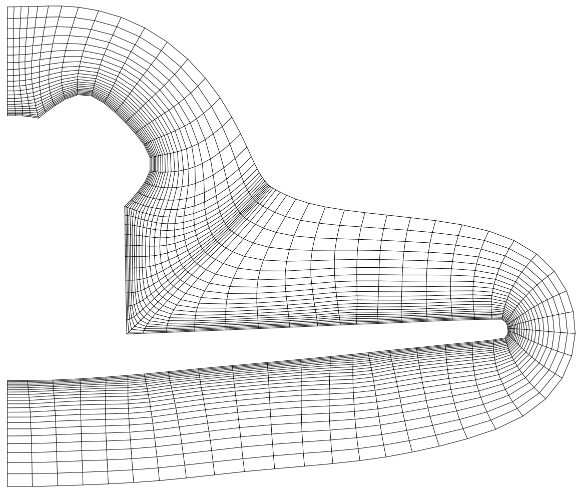

(e) TIMJ=3, ITSVOL=25, SMU2=0.7

Figure 2.4. Example showing the effects of TIMJ, ITSVOL and SMU2.

2.3.5. Basic Rules on the Construction of a Good Surface Grid

The quality of the 3-D volume grid obtained by marching from a surface is directly affected by the quality of the surface grid. The rules described below should be followed as closely as possible when constructing the surface grid.

(1) Large and sudden jumps in the surface grid spacing should be avoided. A stretching ratio of 1.0–1.3 is best, 1.3–3.0 is acceptable, 3.0 or above is not recommended.

(2) The grid spacings on each side of a convex or concave corner should be about equal. The sharper the corner, the more important it is to have equal grid spacings on each side. Not only is this grid property desirable for producing smooth volume grids but it is essential for satisfactorily resolving the flow around the corner by the flow solver.

(3) The sharper the angle of a convex corner, the more important it is to cluster points towards the corner which helps the volume grid to achieve good resolution in this region. On the other hand, the sharper the angle of a concave corner, the more important it is to *not* cluster into the corner. A uniform or declustered surface grid spacing at the concave corner can significantly ease the generation of the volume grid.

2.4. Grid Generation and Quality Problems

Some commonly encountered problems when using *HYPGEN* and the built-in grid quality checks are described in this section.

2.4.1. Suggested Remedies for Commonly Encountered Problems

The following is a list of commonly encountered problems and suggested remedies when using *HYPGEN*.

(1) If the code aborts before a grid is generated, it usually means the user-supplied surface grid may be of the wrong format or it is incompatible with the specified boundary conditions in the input parameters file.

(2) If the 3-D grid appears completely distorted, first check that the surface grid looks fine, that the J, K and L directions form a right-handed set and that all the boundary conditions are consistent with what is intended before proceeding to adjust the grid smoothing parameters.

(3) If there are negative cell volumes and/or Jacobians present in the 3-D grid, first try to adjust SMU2 and/or ITSVOL. The problem can usually be fixed by adjusting these two parameters. For example, negative Jacobians in the far field can usually be fixed by increasing SMU2. A highly concave region often needs large SMU2 (greater than ≈ 2) and large ITSVOL (greater than ≈ 50). However, large values of SMU2 and/or ITSVOL can ruin the grid smoothness at sharp convex corners such as an O-grid wing trailing edge.

(4) If negative cell volumes and/or Jacobians persist after many attempts of different combinations of grid smoothing parameters, the surface grid may possess

features requiring special treatments that are beyond the ability of the current scheme. The problem can sometimes be solved by modifying the surface grid through redistribution of points or refinement of difficult regions, e.g., rounding a very sharp convex or concave corner (also see §2.3.5 for some guidelines on the construction of a good surface grid).

2.4.2. Grid Quality Checks

The code performs two grid quality checks every time a grid is generated. The first and more stringent test is a cell volume computation by tetrahedron decomposition [6]. The second is a Jacobian computation using the same definition as that of the flow solver *OVERFLOW* [7]. If a cell passes the second test, it will definitely run through the flow solver. However, if a cell passes the second test but not the first, it is probably distorted in some way and accuracy may be degraded in this region.

The earlier version of the Jacobian computation in *OVERFLOW* and *HYPGEN* was based on a finite volume algorithm [8,9]. Asymmetric Jacobians for a symmetric grid can result from the above algorithm due to the usage of only one of the diagonals of the cell for the Jacobian computation. Although the error is small (typically in the 9th decimal place or smaller on the Cray), it may still cause errors in very fine meshes. A remedy has been implemented into the latest versions of *OVERFLOW* (version 1.6ab or later) and *HYPGEN* (version 1.3 or later) by averaging the four Jacobians computed using the four different diagonals of the cell. Symmetric Jacobians are obtained down to round off error. Unless the cells are very skewed, the values of the Jacobians computed using the current symmetric method should be very close to the values computed using the previous asymmetric method.

2.5. Code Compilation and Execution

The *HYPGEN* package includes: the source codes, a library of examples and PostScript files containing this documentation. The files in the package are:

makehypgen	- command file for compiling *HYPGEN* on the CRAY Y-MP
man.ps*	- PostScript files containing this documentation (*=1-6)
hypgen.f	- Fortran source code of *HYPGEN*
setzeta.f	- Fortran source code of tool for creating variable far field, initial and end grid spacings input file
convi.f	- Fortran source code of tool for converting input parameters file from version 1.3 or earlier to that of version 2.0
examples	- Subdirectory containing a library of examples

HYPGEN is written in Fortran 77 and can be compiled and run on any machine with a Fortran 77 compiler. Double precision may be needed on 32-bit machines when the ratio of a typical length of the entire surface grid to the smallest grid spacing is larger than $\approx 10^7$, e.g., the flight Reynolds number grids for the Space Shuttle where the normalized length of the vehicle is $O(1)$ and the initial grid spacing is 1.6×10^{-7}.

On SGI workstations, *hypgen.f* can be compiled as follows:

```
f77 -o hypgen hypgen.f
```

Further optimization can be invoked by using the `-O` option (same as `-O2`) which may result in a factor of 2 or more speed up in code execution. However, certain versions of `f77` may produce erroneous code for *hypgen.f* resulting in a core dump with the `-O` option. In that case, the `-O1` (default) option that does less optimization should be used.

On the CRAY Y-MP, CRAY-2 or CRAY C-90, further optimization (with only slight speed improvement) can be achieved by compiling *hypgen.f* as follows:

```
cf77 -Zv -Wd″-e78″ -o hypgen hypgen.f
```

Under a UNIX operating system, the code can then be executed by typing

`hypgen` < *input_parameters_filename* > *output_messages_filename*

The maximum grid dimensions are specified at the top of the main program where M3D = max number of points in the 3-D grid, M2D = max number of points in a J-K plane and M1D = max number of points in any one direction. At least two extra points in each direction should be allowed when computing M1D, M2D and M3D. An error is reported if the input grid dimensions exceed the ones declared in the program. If this occurs, the code should be recompiled with the appropriate M1D, M2D and M3D.

An Orbiter grid with dimensions $98 \times 77 \times 57$ (= 430122 pts) was used as a test case to compare the CPU times on different machines (see Table 2.1). Close to a hundred thousand grid points per second is achieved on the CRAY Y-MP. Approximately 20% of the CPU time indicated is spent on grid quality checks. A factor of 3 speed up is achieved by compiling with the `-O` option on the SGI Indigo 2 for this case.

	CPU time	MFLOPS	CPU time / grid pt.	No. of grid pts. / CPU sec.
CRAY Y-MP	$4.5s$	141	$10.5 \times 10^{-6}s$	0.95×10^6
CRAY C-90	$1.9s$	339	$4.4 \times 10^{-6}s$	2.27×10^6
SGI Indigo 2 *	$220.0s$	3	$5.1 \times 10^{-4}s$	2.0×10^3
SGI Indigo 2 †	$71.6s$	9	$1.7 \times 10^{-4}s$	6.0×10^3

Table 2.1. Comparison of CPU times for different machines for Orbiter test case (* compiled with default option, † compiled with `-O` option; MFLOP for the Indigo 2 is computed from an estimate of the operation counts on the CRAYs).

2.6. Illustrated Examples

The examples directory contains a collection of subdirectories, each containing one or more examples related to the name of that subdirectory. For each example, a surface grid in a formatted *PLOT3D* file (*surf.i*), a sample input parameters file and a *PLOT3D* command file for viewing purposes are provided. The input file for *setzeta.f* is also given for cases that use a variable far field distance or initial/end grid spacings. Brief discussions on each of the examples are given below.

(1) **Ellipsoid**

One of the simplest examples in the collection is an ellipsoid grid shown in Figure 2.5a. The grid is periodic in K and has singular axis points at the two boundaries in J. Figure 2.5b shows a vertical strut grid with the same grid topology as the ellipsoid. Although the vertical strut contains some concave and convex corners, the same set of input parameters is used to generate the ellipsoid and the vertical strut.

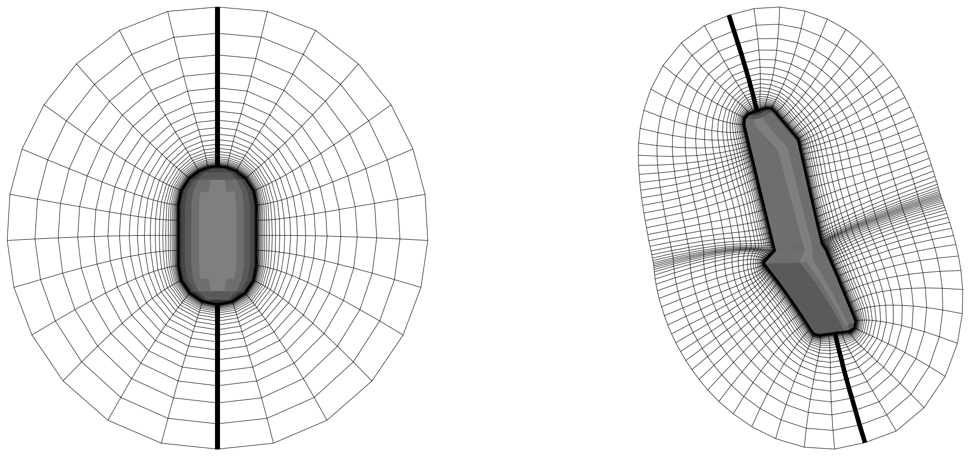


Figure 2.5. Two K = constant slices through the volume grid of (a) an ellipsoid, (b) a vertical strut (━━ singular axis line at J = 1 and $jmax$).

(2) **Corner**

The parameters used to generate grids out of concave corners are demonstrated by the two examples in the corner directory: a concave corner at 20 degrees and a double concave corner (see Figure 2.6). A fairly large TIM factor is used in the direction of the corner together with a relatively large smoothing coefficient SMU2. Also, more space is made available for the grid to march out by splaying the side boundaries.

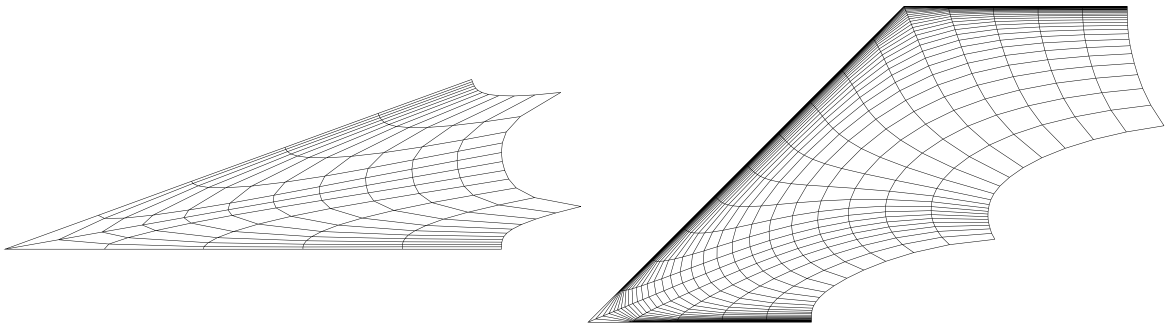
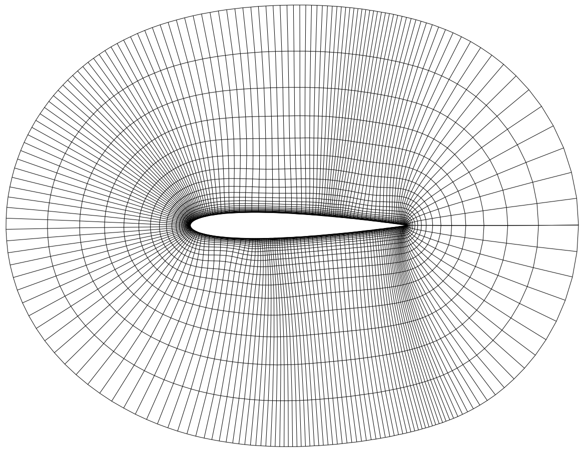


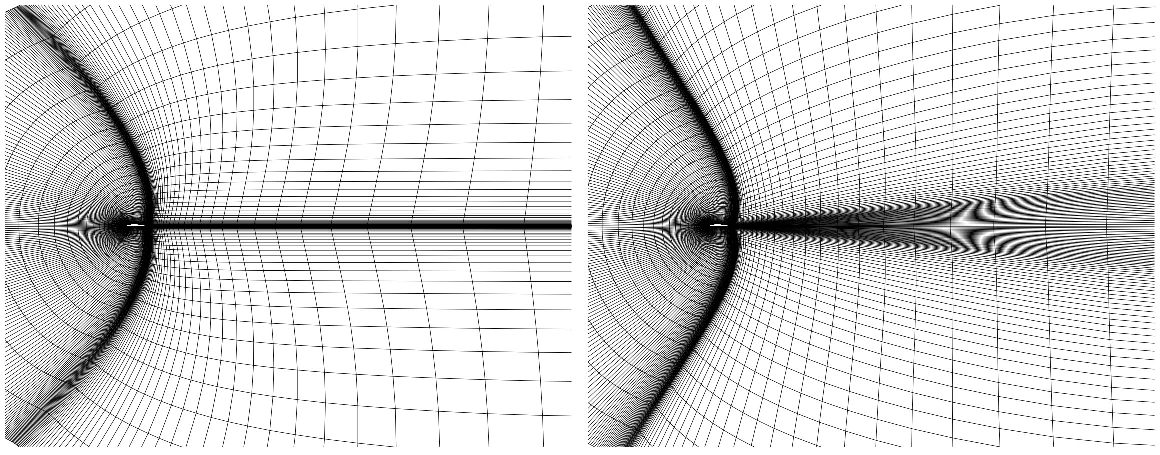
Figure 2.6. A slice through the volume grid of (a) a 20-degree concave corner, (b) a double concave corner.

(3) **Airfoil**

Parameters used to generate different types of airfoil grids are demonstrated in the airfoil directory. These include: an O-grid, a C-grid with constant initial spacing everywhere and a C-grid with two L-regions and a variable initial spacing and far field distance in the first L-region along the wake cut (see Figure 2.7). Figure 2.7c shows that both the far field distance and the initial normal grid spacing along the wake cut in the L-region close to the body surface are made to increase linearly with distance downstream, resulting in a fanning of the grid points in the wake. In order to reduce grid skewness, two L-regions are employed with a variable far field in the first L-region to limit the growth of the grid in the normal direction. If this limit is absent, the grid lines would tend to bend too far forward resulting in an unsatisfactorily skewed grid.



(a)

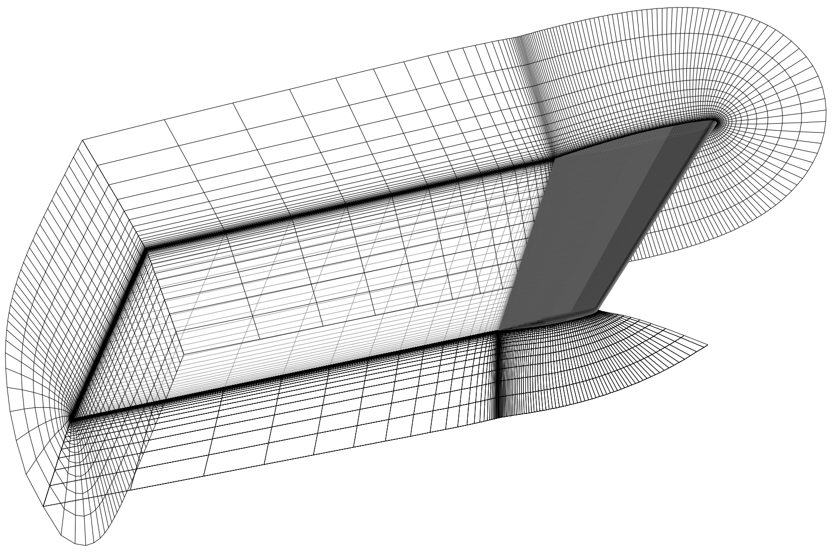


(b) (c)

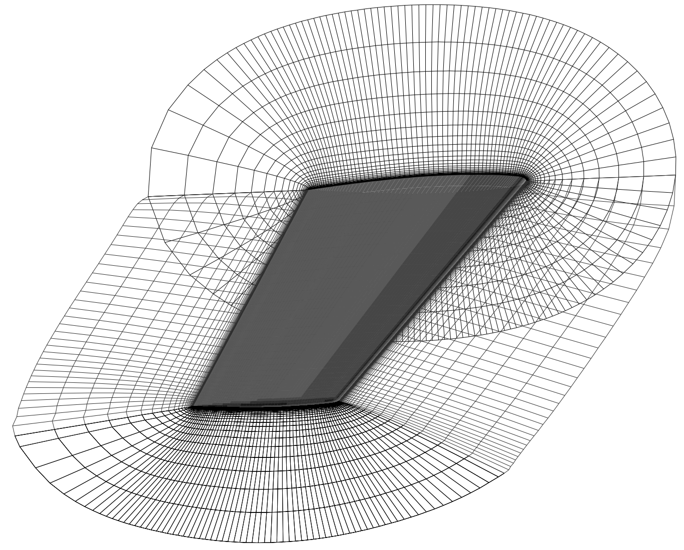
Figure 2.7. (a) Airfoil O-mesh, (b) airfoil C-mesh with single L-region, constant initial spacing, (c) airfoil C-mesh with two L-regions, variable initial spacing and far field distance in first region.

(4) **Wing**

Two examples are given in the wing directory for a single grid over a wing with a closed tip. In the first example, a C-mesh topology is used at each section from the wing root to the wing tip (Figure 2.8a). At the tip, the "C" collapses to a slit where the boundary condition for a floating collapsed edge with matching upper and lower sides (IBC = 7) is applied. In the second example, the C-type grid lines are replaced by O-type grid lines (Figure 2.8b). The O-topology case is more difficult to generate due to the presence of the sharp trailing edge at the collapsed tip.



(a)



(b)

Figure 2.8. Various computational planes of the volume grid for a wing with a collapsed edge at the tip, (a) C-topology, (b) O-topology.

(5) **Fuselage**

Two different boundary condition types and their effects on the volume grid are illustrated by this example for a fuselage section. The first example shows a constant x-plane boundary condition (IBC = 1) applied to the two streamwise x-boundaries (Figure 2.9a). Note that the volume grid is orthogonal to the body surface except near the constant x-plane boundaries where the grid generator detects that some non-orthogonality has to be introduced in order to prevent grid lines from crossing the imposed constant plane. The second example shows the grid obtained by imposing constant interior x-planes at each streamwise station (IBC = 21). Since each slice of the grid is generated as a 2-D slice independently from each other, the volume grid is no longer orthogonal to the body surface (Figure 2.9b).

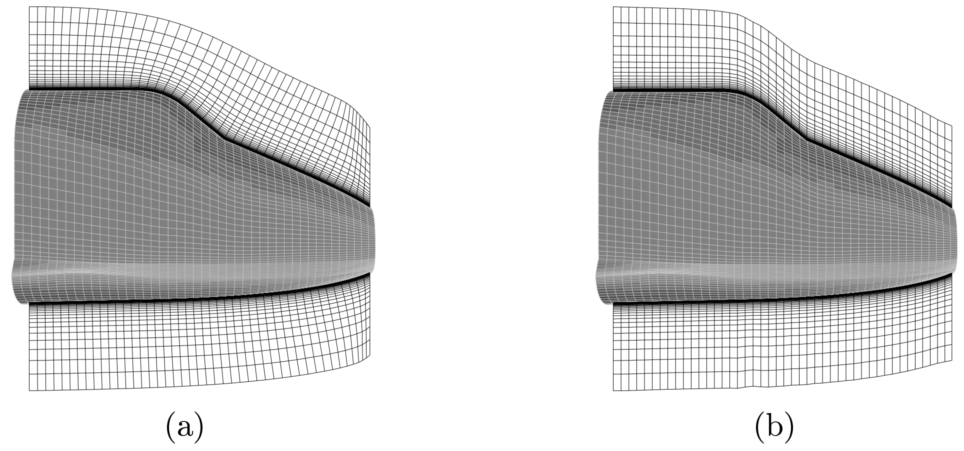


Figure 2.9. Slices through the volume grid of a fuselage section with (a) constant x-planes at streamwise boundaries only, (b) constant x-planes at all streamwise stations.

(6) **Duct**

HYPGEN has some limited capability to do a class of internal grids where one family of grid lines is periodic as in the duct example (Figure 2.10). The cross sections can have variable shapes and radii but it is advantageous to have the grid lines as orthogonal as possible to any bends in the duct. A variable far field distance is used to place the far field shell close to the centerline of the duct. Since the actual far field distance of the grid is dependent on the smoothing present, some trial and error is needed to determine a good value of the distance scale factor. A strategy to employ is to use more points in L than may be needed and then discard the last few L-shells containing bad cells due to overshoot. By the nature of its formulation, a hyperbolic scheme cannot close the grid at the last L-shell as shown in Figure 2.10. A post-processing code can be used to close the grid to a point by averaging the points on the last L-shell.

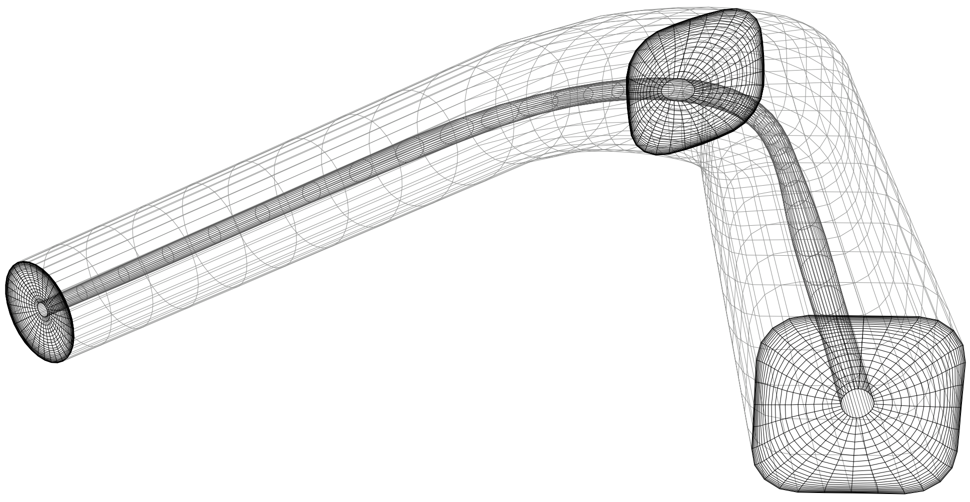


Figure 2.10. Various computational planes of the volume grid for a duct.

(7) **Tank**

This example demonstrates the use of multiple L-regions on the external tank of the Space Shuttle (Figure 2.11). The three L-regions include a stretched viscous region next to the body, an approximately uniformly spaced region farther out and a stretched outer region to take the grid to the far field. The initial grid spacings of the outer two regions are not specified in the input parameters file indicating to the code that the final spacing from an inner region is to be used as the initial spacing of the outer region.

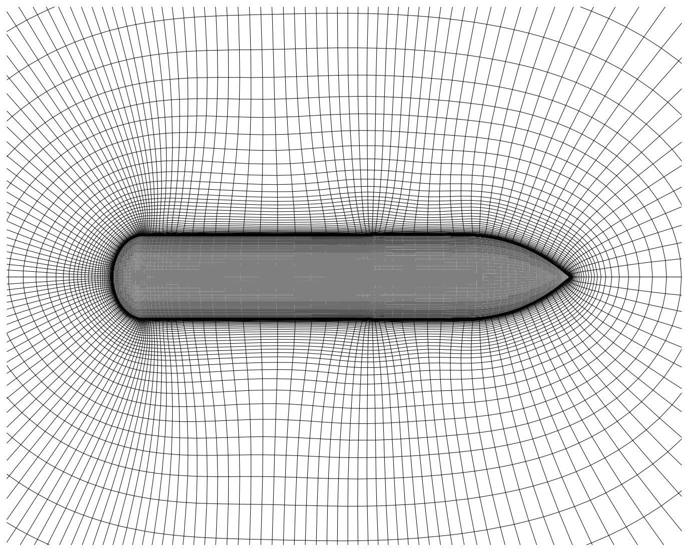


Figure 2.11. Symmetry plane of the volume grid for the external tank showing the three L-regions.

(8) **Orbiter**

In Figure 2.12, the Orbiter is enclosed by a single volume grid. This example illustrates that a good volume grid can be generated over a fairly complex surface grid that includes both the fuselage and the wing.

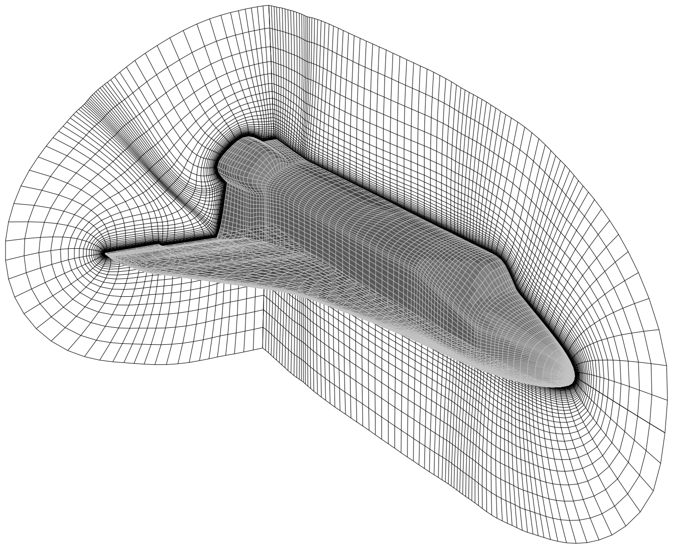


Figure 2.12. Various computational planes of the volume grid for the Orbiter.

(9) **Tail**

This example shows the vertical tail grid with a thin cross section and a singular axis point at the tip (Figure 2.13). The topology is similar to that of a wing. For certain types of wing shapes, it is more convenient to close off the tip with a singular point rather than with a slit as in the wing examples above.

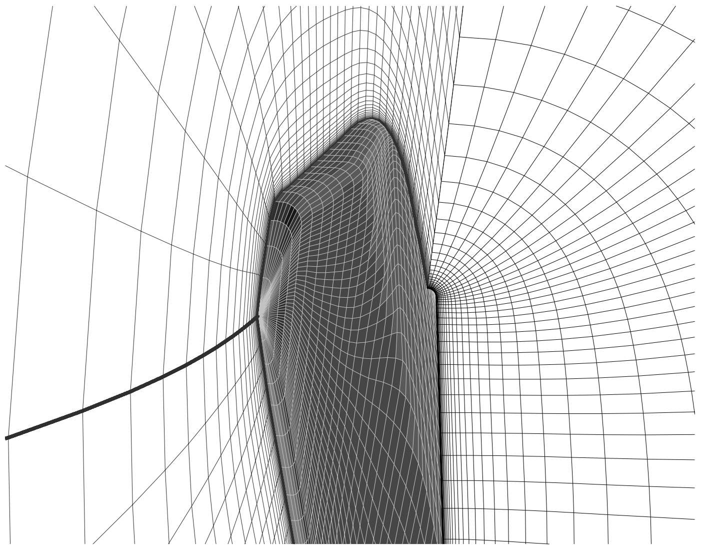
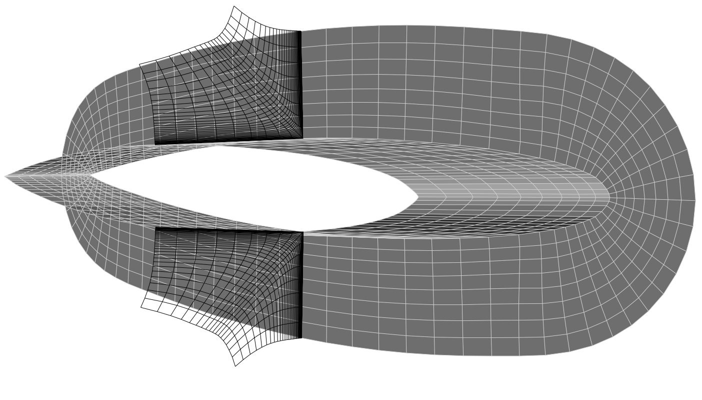


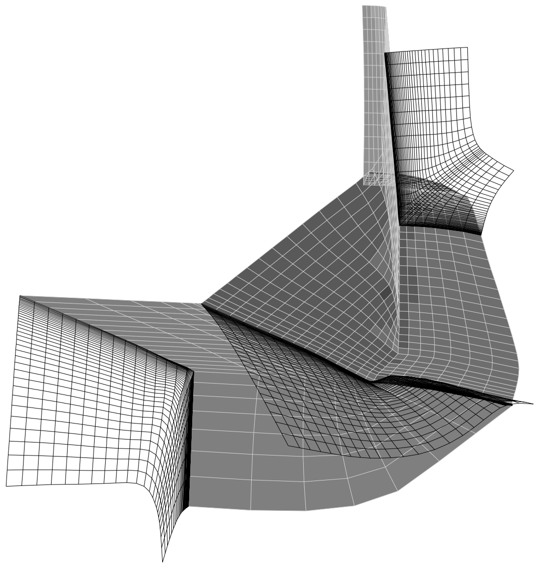
Figure 2.13. Various computational planes of the volume grid near the tip of the vertical tail of the Orbiter (—— singular axis line).

(10) **Collar**

Collar grids [10] are used at the junction of intersecting geometric components to provide proper interpolation for the Chimera overset-grid scheme. The first collar example shows the collar grid for a wing/fuselage junction (Figure 2.14a) while the second collar example shows the collar grid for the vertical tail/Orbiter fuselage junction (Figure 2.14b). These examples demonstrate the parameters used for cases where there are combinations of concave and convex corners on the surface. Note that SMU2 cannot be made too high due to the presence of the convex corners. In cases with both severe concave and convex corners, it may not be possible for *HYPGEN* to produce a satisfactory grid. One remedy is to round off the convex corners so that SMU2 can be increased sufficiently to march the grid out of the concave corners. Redistribution of grid points on the surface may also help (see §2.3.5).



(a)



(b)

Figure 2.14. Various computational planes of the volume grid for (a) a wing/fuselage collar grid, (b) the vertical tail/Orbiter fuselage collar grid.

(11) **Feedline**

This example shows slices through the volume grid of the liquid hydrogen feedline of the Space Shuttle attach hardware (Figure 2.15). The grid is able to grow out of a complex combination of concave and convex corners on the surface. A similar level of geometric complexity is present in a number of components of the current grid system for the Space Shuttle Launch Configuration [11] (Orbiter, External Tank, Solid Rocket Boosters and attach hardware) where most of the volume grids have been generated using *HYPGEN*.

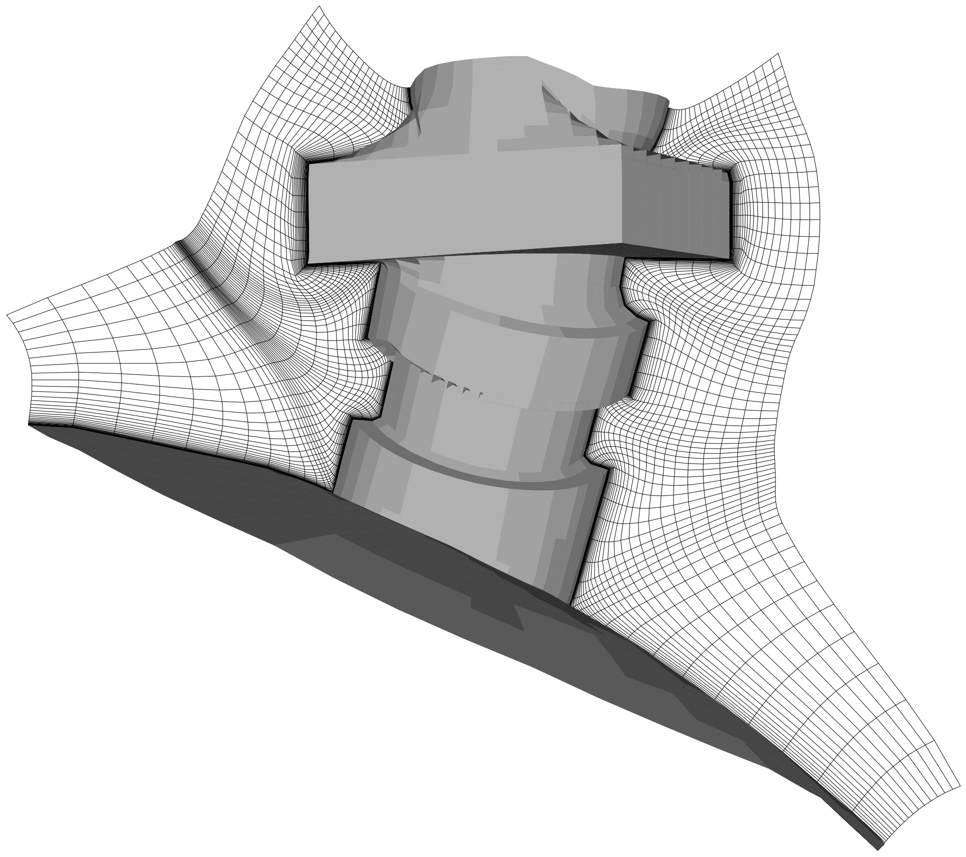
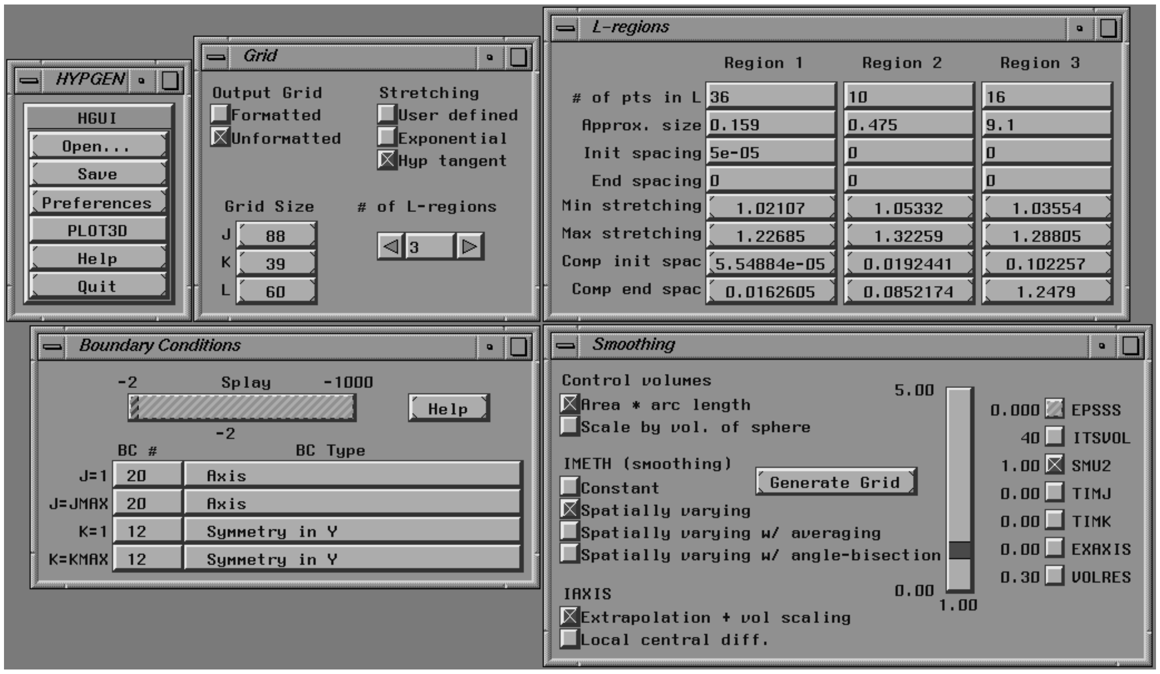


Figure 2.15. Various computational planes of the volume grid for the liquid hydrogen feedline of the Space Shuttle attach hardware.

3. HGUI Graphical User Interface

3.1. Introduction to the *HGUI* Graphical User Interface

HGUI (stands for *HYPGEN* Graphical User Interface) is a graphics front end using Silicon Graphics (SGI) workstations for the hyperbolic grid generator *HYPGEN* developed by Chan and Steger [3]. It uses the NAS Panel Library [12] and provides menus, buttons, sliders and typein fields (see Figure 3.1) for users to enter parameters needed to execute *HYPGEN*. A hook to *PLOT3D* is built in to allow viewing of the surface and volume grids; however, users still need to type in the filename of the *PLOT3D* "com" file written by *HGUI* to display the grids. The current version of *HGUI* allows *HYPGEN* to run on a remote machine through the use of shell scripts on UNIX operating systems. The volume grid generated is then copied back to the local machine for graphics manipulation.



Grid

Output Grid: Formatted [✓ Unformatted]

Stretching: User defined, Exponential, [✓ Hyp tangent]

Grid Size: J 88, K 39, L 60

# of L-regions: 3

L-regions

	Region 1	Region 2	Region 3
# of pts in L	36	10	16
Approx. size	0.159	0.475	9.1
Init spacing	5e-05	0	0
End spacing	0	0	0
Min stretching	1.02107	1.05332	1.03554
Max stretching	1.22685	1.32259	1.28805
Comp init spac	5.54884e-05	0.0192441	0.102257
Comp end spac	0.0162605	0.0852174	1.2479

Boundary Conditions

Splay: -2 … -1000, value -2

	BC #	BC Type
J=1	20	Axis
J=JMAX	20	Axis
K=1	12	Symmetry in Y
K=KMAX	12	Symmetry in Y

Smoothing

Control volumes: [✓ Area * arc length] Scale by vol. of sphere

IMETH (smoothing): Constant [✓ Spatially varying] Spatially varying w/ averaging, Spatially varying w/ angle-bisection

IAXIS: [✓ Extrapolation + vol scaling] Local central diff.

Slider: 5.00 … 0.00, value 1.00

0.000	EPSSS
40	ITSVOL
1.00	[✓ SMU2]
0.00	TIMJ
0.00	TIMK
0.00	EXAXIS
0.30	VOLRES

Figure 3.1. Various panels of the *HYPGEN* Graphical User Interface *HGUI* .

3.1.1. Installation

The installation instructions below are particular to the CRAY Y-MP as the remote machine. However, the same installation procedure can be applied to any remote computer as long as programs are available to convert SGI unformatted files to unformatted files for the remote computer, and vice versa.

On an SGI workstation:

1. `zcat hgui.tar.Z | tar xfvo -`
2. `cd hgui/bin` (change directory to *hgui/bin*)
3. `cp hypgen.sgi hgui` *user's_bin_directory* (e.g., *user's_bin_directory* can be `/u/wk/my_name/bin`)
4. Compile *hypgen.f* (source code in *hgui/hypgen*) and move the executable *hypgen* to the same directory as *hgui* and *hypgen.sgi*.
5. Make sure that *hypgen.sgi*, *hgui* and *hypgen* are in a directory specified in the environment variable "path" for C-shell (*csh*) or "PATH" for Bourne shell (*sh*).

On a Cray Y-MP (skip this part if grids are to be generated on an SGI workstation):

1. Compile *hypgen.f* and *zconv.F* (source code in *hgui/hypgen* and *hgui/tools*, respectively) to produce the executables *hypgen* and *zconv*.
2. Move *hypgen*, *hypgen.cray* (in *hgui/bin*) and *zconv* to a directory along the command searching path specified by the environment variable "path" for *csh* or "PATH" for *sh*.
3. Check if *4d2cray* and *4Dconv* (programs used by shell script *hypgen.cray* to convert SGI unformatted to/from Cray unformatted files) are available or in a directory specified in the environment variable "path" for *csh* or "PATH" for *sh*. If *hypgen* is to be run on a different remote computer, replace *4d2cray* and *4Dconv* with programs that do the file format conversion between the workstation and the remote computer and modify the shell script file, *hypgen.cray*, to reflect this change.
4. For *csh* users, add/modify the "path" environment variable in *.cshrc* to include the directory where *hypgen.cray* and *hypgen* are located. Usually, "path" is set in *.login* which is not "sourced" by the remote shell *rsh*. Thus "path" has to be set in *.cshrc*; otherwise, the absolute path, e.g.,

   ```
   /u/ne/my_name/bin/hypgen.cray,
   ```

   has to be used instead of *hypgen.cray*. The following shows an example of the syntax for setting "path" in *.cshrc*.

   ```
   set path =( .  ~/bin /usr/bin /bin /usr/local/bin \
   /usr/unsupported/bin )
   ```

   For *sh* users, add/modify the "PATH" environment variable in *.profile* accordingly. The syntax is like

   ```
   PATH=.:$HOME/bin:/usr/bin:/bin:/usr/local/bin:
   ```

5. Edit the *.rhosts* file on the workstation and the remote computer to include entries for both the workstation and the remote computer. The syntax of the entry in the *.rhosts* file is best explained by an example:

   ```
   wk000.nas.nasa.gov
   ```

If *HGUI* is set up following the procedures above and problems are still encountered in running *HYPGEN* on a remote machine, the "Answers to Frequently Asked Questions" section (§3.4.2) might contain solutions to the problems.

3.1.2. Startup

HGUI can be started by typing `hgui`. The *"Filename"* panel will then pop up (see §3.2.1), followed by the *"Grid," "L-regions," "Boundary Conditions"* and *"Smoothing"* panels and a *PLOT3D* window (see §3.3). Alternatively, *HGUI* can be started by typing `hgui` *basename*. Then, the *"Filename"* panel will be bypassed and the other panels and the *PLOT3D* window will appear immediately.

The *basename* convention (*basename* followed by an extension) allows easy specification of filenames of various input and output files for running *HYPGEN* using *HGUI* . The input and output files are assumed to have the names listed below when the *basename* option is used for running *HGUI* (see §3.2.1). The filenames used for running *HYPGEN* in batch mode are included in brackets (see §2.2).

basename.i	:	*HYPGEN* input parameters file
basename.out	:	*HYPGEN* output messages file
basename2d.dat	:	surface grid file (*surf.i*)
basename3d.dat	:	volume grid file (*plot3d.dat*)
basename.com	:	*PLOT3D* command file
basenamevar.dat	:	variable far field, initial/end grid spacings file (*zetavar.i*)
basenamestr.dat	:	user-defined stretching function file (*zetastr.i*)

The *basename2d.dat* file is required for input. If the *basename.i* file exists, *HGUI* will read the input parameters from this file. In any case, the input parameters can be selected and modified inside *HGUI* and written out to the *basename.i* file.

3.1.3. Objects of the Graphical User Interface

The various objects of the graphical user interface (Figure 3.2) are described below.

Using the mouse - Only the left mouse button is used with *HGUI* to operate on menus, buttons, sliders, and typeins. The following explains the terms used for mouse operations in this manual.

Click - Press the left mouse button.

Drag - Move the mouse while holding down the left mouse button.

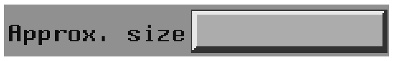
Typein - This is used to accept input from users. The object looks similar to wide buttons, except that the wide buttons have beveled edges. In order to make changes take effect immediately for typein fields, the changes have to be entered followed by a carriage return.

Slider - The current value of the slider is controlled and reflected by the position of a slider bar within a bounding rectangular region. The min/max values of a slider are shown along the side of the slider. To change the value of the slider, click within the rectangular region. The slider bar jumps to the mouse location and then follows subsequent mouse motion until the mouse button is released. One can also click on the slider bar, then drag it to change the value of the slider. To enter slider fine control mode, press the control key on the keyboard while holding down the left mouse button. This is useful when the difference of the min/max values is large.

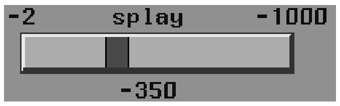
Radio button - Usually several radio buttons form a group within which they interact with each other. When a radio button is selected (i.e., clicked with left mouse button), the rest of the radio buttons in the same group are unselected.

Wide button - When a wide button is clicked, it is used to perform function as labeled on the button or to display related information. A wide button will stay highlighted through out the entire period of the action requested.

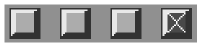
Menu - This offers ways to access different parts of *HGUI* . Submenus become active by dragging the mouse (i.e., moving with left mouse button down) over them.



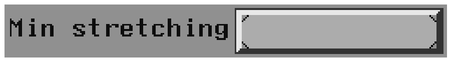
(a) Typein



(b) Slider



(c) Radio button



(d) Wide button

Figure 3.2. Objects of the graphical user interface.

3.2. Main Menu and its Associated Panels and Submenus

The main menu (Figure 3.3) provides entry points for performing file I/O and specifying various customizations.

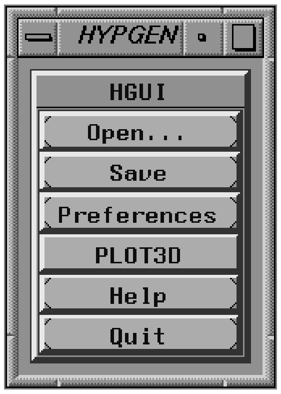


Figure 3.3. Main menu.

Open	: Pops Filename panel for file I/O.
Save	: Saves *HYPGEN* input parameters to a file which can be of a different name than that specified by the basename.
Preferences	: Pops Preferences panel to allow choice of location of *hypgen* or shell script to run *hypgen*, customizations of *PLOT3D* popup window and the range of allowable values for smoothing parameters.
PLOT3D	: Pops submenu for *PLOT3D* related options.
Help	: Provides help information on menus.
Quit	: Exit *HGUI* .

3.2.1. Filename Panel

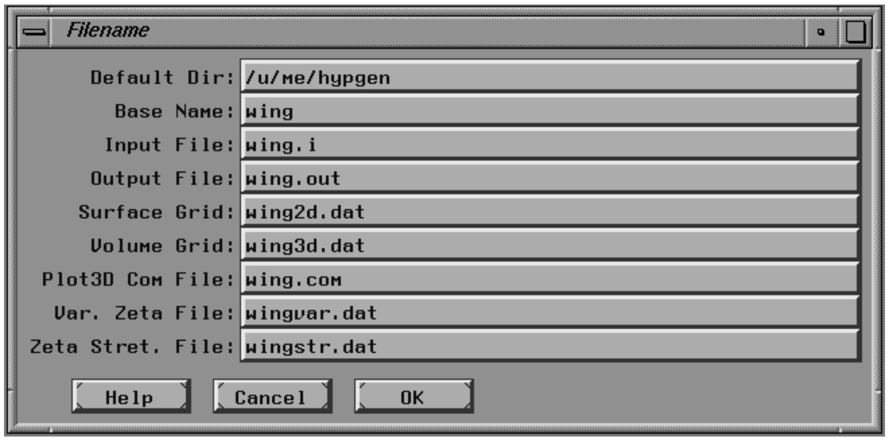
When the *"Open"* button in the main menu is clicked, the *"Filename"* panel as shown in Figure 3.4 is popped up front providing typein fields to enter filenames for files needed to run *HYPGEN* using *HGUI* . The *"Default Dir"* typein field shows the current working directory and can be changed. However, if a directory specified cannot be found, the directory is not changed and a warning panel showing the error message will pop up. Whenever a file cannot be found by *HGUI* , a similar warning message will appear. The filenames shown in the *"Filename"* panel are used by *HGUI* for its input and output files (see Table 1.2 in §1 for the required and optional files). The *"Base Name"* typein field allows easy entry for the filenames of the various files. As shown in Figure 3.4, *HGUI* appends proper filename extensions to what is entered in the *"Base Name"* typein field.

The default filenames used by *HGUI* for the input and output files are given below.

basename.i	:	*HYPGEN* input parameters file
basename.out	:	*HYPGEN* output messages file
basename2d.dat	:	surface grid file (*surf.i*)
basename3d.dat	:	volume grid file (*plot3d.dat*)
basename.com	:	*PLOT3D* command file
basenamevar.dat	:	variable far field, initial/end grid spacings file (*zetavar.i*)
basenamestr.dat	:	user-defined stretching function file (*zetastr.i*)

Further modification of individual filenames is possible by manually changing any of the typein fields.

The *"Help"* button in the panel gives help information, the *"Cancel"* button discards the filenames entered, and the *"OK"* button accepts the filenames entered and performs the following functions. The min/max coordinates of the surface grid are computed and saved to a *PLOT3D* com file and *PLOT3D* will be launched in a separate window if it is not already running. Then the panels for entering/modifying *HYPGEN* input data (see §3.3) will pop up in place of the *"Filename"* panel. The surface grid can be viewed by executing the filename entered in the *" PLOT3D Com File"* typein field in the *PLOT3D* window.


Filename

Default Dir:	/u/me/hypgen
Base Name:	wing
Input File:	wing.i
Output File:	wing.out
Surface Grid:	wing2d.dat
Volume Grid:	wing3d.dat
Plot3D Com File:	wing.com
Var. Zeta File:	wingvar.dat
Zeta Stret. File:	wingstr.dat

Figure 3.4. Filename panel.

3.2.2. Preferences Panel

The *"Preferences"* panel (as shown in Figure 3.5) provides typein fields for dynamically changing the filename of the executable for *HYPGEN* and customization of the window shell to launch *PLOT3D*. There are also buttons for saving and reading preferences to/from a file called *.hguirc*. At startup, *HGUI* searches for *.hguirc* in the current working directory. If *.hguirc* cannot be found there, it then looks into the user's home directory for the file. The following shows what is in a typical *.hguirc* file:

```
hypgen
wsh -r1000 -m80,24 -p530,0 -fScreen-Bold.15 -C2,0,3,2 -c csh -c \
"source ~/.login; plot3d"
```

The first line specifies the filename of the executable for *HYPGEN* , and the second line specifies the user's window shell. If *.hguirc* contains only one line (i.e., the window shell is not specified), the window shell shown above will be used as the default to launch *PLOT3D*.

If there is no *.hguirc* in the current working directory and the user's home directory, *HGUI* then checks the environment variables HYPGEN and P3DWIN, for the preference settings. The following shows the syntax for setting the environment variables HYPGEN and P3DWIN.

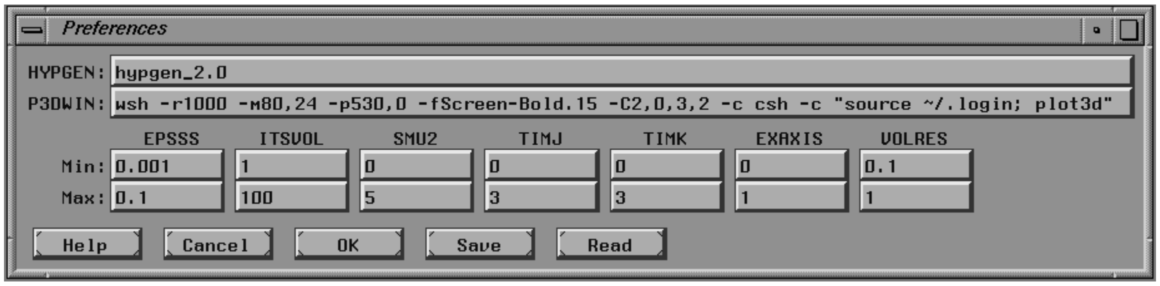
```
setenv HYPGEN hypgen
setenv P3DWIN "xterm -fn courier-bold.14 -fg green -bg black -e plot3d"
```

The P3DWIN shown above uses xterm as the window shell for launching *PLOT3D* with courier-bold.14 as the character font, black as the background and green as the foreground.

If *.hguirc* is not available and the environment variables HYPGEN and P3DWIN have not been set, then the information in the *.hguirc* example above will be used as the default.

The first line of *.hguirc* or the environment variable HYPGEN can also be set to a shell script with all the necessary arguments to run the grid generator on a remote machine (usually a faster one). More details are given in §3.1.1 and §3.4.1 on how to set this up.

The *"Min/Max"* typein fields are provided to set minimum and maximum allowable values for the smoothing parameters in the *"Smoothing"* panel. The *"Help"* button provides relevant information about customization of user preferences. The *"Cancel"* button discards all changes made not followed by a carriage return, and the *"OK"* button accepts all changes shown. The *"Save"* button saves preferences to *.hguirc* in the current working directory, and the *"Read"* button reads in preferences from *.hguirc* in the current working directory or in the user's home directory.



HYPGEN: hypgen_2.0

P3DWIN: wsh -r1000 -m80,24 -p530,0 -fScreen-Bold.15 -C2,0,3,2 -c csh -c "source ~/.login; plot3d"

	EPSSS	ITSVOL	SMU2	TIMJ	TIMK	EXAXIS	VOLRES
Min:	0.001	1	0	0	0	0	0.1
Max:	0.1	100	5	3	3	1	1

Figure 3.5. Preferences panel.

3.2.3. PLOT3D Submenu

The *PLOT3D* submenu (Figure 3.6) is accessed by selecting *"PLOT3D"* from the main menu. The following functions are available:

New PLOT3D : Pops a new shell for *PLOT3D*.

Rehash : If a *PLOT3D* command file is moved into the default directory indicated in the *"Filename"* panel after *HGUI* is initiated, a rehash is necessary to make the command file accessible by *PLOT3D*. If the default directory in the *"Filename"* panel is changed, a rehash is performed automatically.

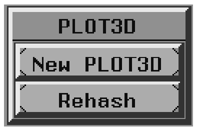


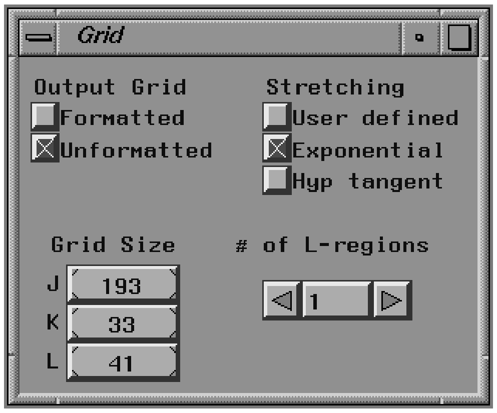
Figure 3.6. PLOT3D submenu.

3.3. Input Panels

The parameters in the input parameters file for *HYPGEN* can be initialized or modified via the four input panels: *"Grid," "L-regions," "Boundary Conditions"* and *"Smoothing"* which are popped up after the startup for *HGUI* . Each panel is described in more detail below.

3.3.1. Grid Panel

In the *"Grid"* panel (Figure 3.7a), the format of the output volume grid file, the stretching function and the number of regions in the ζ-direction are set by clicking on the corresponding button. The formats of all input files are determined automatically by *HGUI* . The *"Grid Size"* buttons (in Figure 3.7a) are used to show the grid size only and cannot be altered like a typein field.

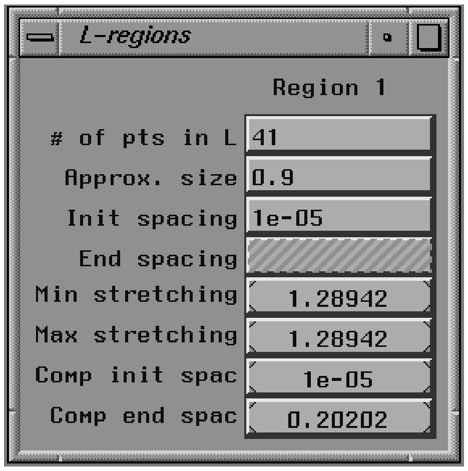

Grid

Output Grid
- Formatted
- [✓ Unformatted]

Stretching
- User defined
- [✓ Exponential]
- Hyp tangent

Grid Size
- J 193
- K 33
- L 41

# of L-regions: 1


L-regions

	Region 1
# of pts in L	41
Approx. size	0.9
Init spacing	1e-05
End spacing	▨
Min stretching	1.28942
Max stretching	1.28942
Comp init spac	1e-05
Comp end spac	0.20202

(a) (b)

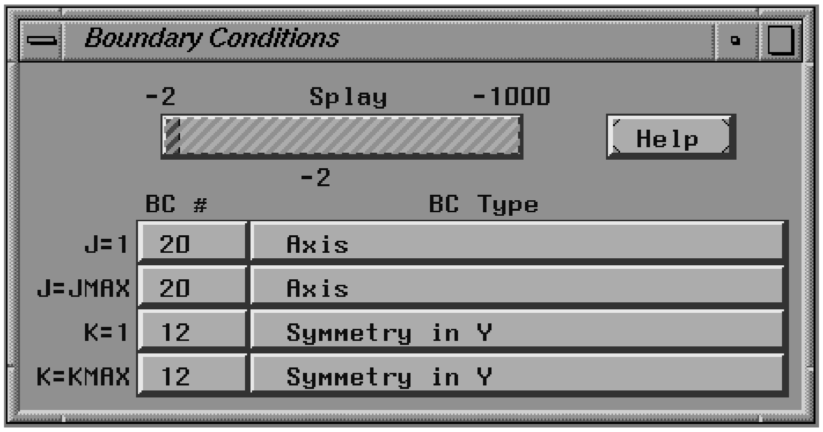
Figure 3.7. (a) Grid panel, (b) L-regions panel.

3.3.2. L-regions Panel

In the *"L-regions"* panel, four typein fields are provided to enter grid related information for each L-region. If more than one L-region is specified, each L-region will have its own column of data. For example, the *"L-regions"* panel in Figure 3.1 shows three L-regions. Any changes to any of the typein fields will affect the min/max stretching ratios and the computed initial/end spacings for all regions in the ζ-direction. The *"End spacing"* typein field in Figure 3.7b is shown with hash pattern meaning that the typein field is not selectable because exponential stretching is chosen (see Figure 3.7a) and the end spacing cannot be specified. The *"End spacing"* typein field will become active when hyperbolic tangent is chosen as the method of stretching. The *"Min/Max stretching"* and *"Computed init/end spacing"* buttons in Figure 3.7b are for showing the corresponding information only and cannot be altered like a typein field. If *"user-defined"* stretching is specified in the *"Grid"* panel, the *"L-regions"* panel will not appear.

3.3.3. Boundary Conditions Panel

The boundary conditions available in *HYPGEN* are given in §2.3.3. Based on the characteristics of the specified surface grid such as periodicity or constant planes, *HGUI* automatically selects an appropriate boundary condition at each edge. A floating boundary condition is chosen when no special features are recognized in the surface grid. Any boundary condition can be changed by the user by clicking on the pop-up menu and then dragging downward over a list of available boundary conditions, or by typing in the boundary condition number in the typein field. A list of available boundary conditions is shown by clicking on the *"Help"* button. The slider (for setting values of splay boundary condition) shown with hash pattern in Figure 3.8 means that it is not selectable and will become active only when the splay boundary condition is chosen.



-2 Splay -1000

-2

	BC #	BC Type
J=1	20	Axis
J=JMAX	20	Axis
K=1	12	Symmetry in Y
K=KMAX	12	Symmetry in Y

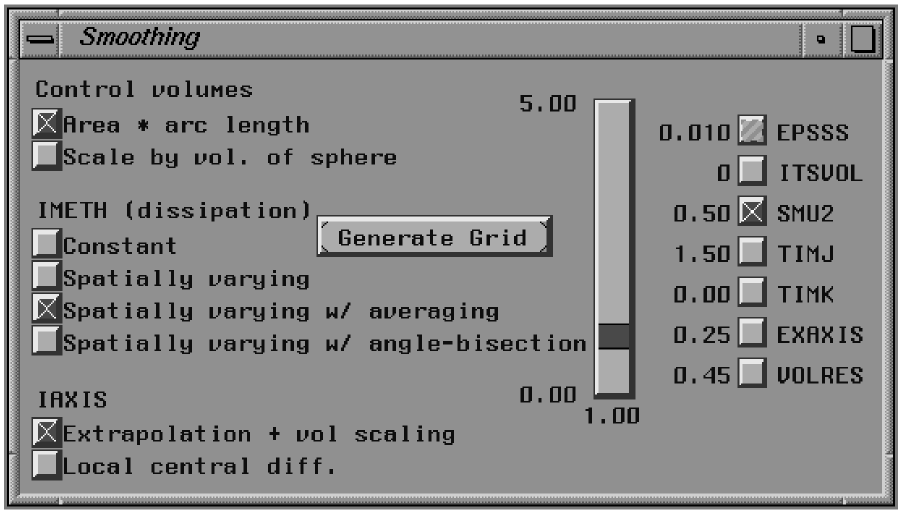
Figure 3.8. Boundary conditions panel.

3.3.4. Smoothing Panel

The *"Smoothing"* panel (Figure 3.9) provides buttons and sliders for choosing options and setting values for smoothing parameters. Explanations of the available options and smoothing parameters are given in §2.3.4. In order to set the values for each parameter, e.g., SMU2, click first on the *"SMU2"* button, then drag the slider bar up or down until the value shown at the bottom of the slider meets the desired value, and then release the mouse button. The label of the *"SMU2"* button will show the value just set. The *"Generate Grid"* button spawns a shell to run the executable of *HYPGEN* specified in the *"Preferences"* panel.

3.4. Running *HYPGEN* on a Remote Machine

The methods and shell scripts used for running *HYPGEN* on a remote machine are described in §3.4.1. Answers to frequently asked questions on these methods are provided in §3.4.2.



Control volumes
[✓ Area * arc length]
Scale by vol. of sphere

IMETH (dissipation)
Constant
Spatially varying
[✓ Spatially varying w/ averaging]
Spatially varying w/ angle-bisection

IAXIS
[✓ Extrapolation + vol scaling]
Local central diff.

0.010 EPSSS
0 ITSVOL
0.50 [✓ SMU2]
1.50 TIMJ
0.00 TIMK
0.25 EXAXIS
0.45 VOLRES

Figure 3.9. Smoothing panel.

3.4.1. Methods and Shell Scripts

A pair of shell scripts, *hypgen.sgi* and *hypgen.cray*, are available in the *HGUI* package that allows *HYPGEN* to be run on the CRAY Y-MP with Cray UNICOS 6.0 and *HGUI* to be run on an SGI workstation with IRIX 4.0. The shell script *hypgen.cray* can be modified to run on any remote machine using the UNIX operating system. Users are advised to read the comment statements in the shell scripts before making modifications. Details on setting up *HGUI* to run *HYPGEN* on the CRAY Y-MP are given in §3.1.1.

HGUI allows several ways for *HYPGEN* to run on a remote computer. The following shows all the possible methods. Users are free to pick what is convenient. In the example below, *hypgen.sgi* is the script to be run on the SGI workstation, "*machine*" is the name of the remote machine, *hypgen.cray* is the script to be run on the remote machine and */scr/my_name/geom* is the directory on the remote machine that *HYPGEN* stores input and output files.

Method 1: Set the environment variable HYPGEN, e.g.,

```
setenv HYPGEN "hypgen.sgi machine hypgen.cray /scr/my_name/geom"
```

Method 2: Put what would be set for the environment variable HYPGEN in the first line of *.hguirc*, e.g.,

```
hypgen.sgi machine hypgen.cray /scr/my_name/geom
```

Method 3: Choose *"Preferences"* from the main menu of *HGUI* and type in what would be given to the environment variable HYPGEN in the *"HYPGEN"* typein field. Then click on the *"OK"* button to confirm the setting. The setting can also be saved to *.hguirc* in the current working directory for later use.

Method 4: Choose *"Preferences"* from the main menu of *HGUI* and click on the *"Read"* button to read from a previously saved *.hguirc* file in the current directory.

Combinations of the above methods can be used. For example, the user can set the environment variable HYPGEN before running *HGUI* and later read from a *.hguirc* file by clicking on the *"Read"* button in the *"Preferences"* panel or type in the new preference in the *"HYPGEN"* typein field.

The examples shown in Methods 1 and 2 are particular to the shell scripts *hypgen.sgi* and *hypgen.cray* that are provided along with the *HGUI* package. User-supplied shell scripts can also be used with the appropriate changes. The four arguments of the shell script are

< *arg 1* > < *arg 2* > < *arg 3* > < *arg 4* >

where

arg 1 = shell script filename

arg 2 = remote computer

arg 3 = remote shell script

arg 4 = remote directory to store input and output files

3.4.2. Answers to Frequently Asked Questions

Q. The environment variable "path" has been set in *.cshrc* on the remote machine to include the directory where *hypgen* and *hypgen.cray* are located, but *HGUI* still will not run *hypgen* and *hypgen.cray* on the remote machine.

A. The following statement probably precedes the statements that set the path in *.cshrc*

```
if (!  $?prompt) exit
```

The above statement would cause *rsh* to exit immediately without proceeding to set the path for *rsh*. Thus, the statements that set the path in *.cshrc* have to be moved before the above statement. One way to check if *rsh* has the proper path set to run *HYPGEN* remotely is to run the following from the SGI workstation,

`rsh` *remote_machine_name* `env`   (AT&T SYSV UNIX)

or

`rsh` *remote_machine_name* `printenv`   (BSD UNIX)

and check the "PATH" variable returned from the above command.

Q. *HGUI* has been set up properly. The *rsh* returns the right path from the remote machine and *HGUI* still does not run *hypgen.cray* and *hypgen* remotely.

A. It is possible that the remote machine does not know the existence of the local workstation. Try the following command from the workstation to see if this is the case.

`rsh` *remote_machine_name* `rsh` *workstation_name*

or if the remote machine is a Cray running UNICOS

`rsh` *remote_machine_name* `remsh` *workstation_name*

If a message like "Unknown host" appears, then the remote machine does not know the existence of the workstation. This can be fixed by assigning the "Local_Domain" variable in *hypgen.sgi* with the domain name of the workstation.

The following example shows what the hostname and the local domain name mean in *hypgen.sgi*:

```
wk000.nas.nasa.gov
```

The hostname is "wk000" and local domain name is "nas.nasa.gov."

Usually, if the local workstation is not on the same domain as the remote machine, e.g., eagle and NAS workstations, then the "Local_Domain" variable in *hypgen.sgi* has to be modified.

Q. The path on the remote machine is set up fine and the remote machine communicates properly with the workstation, but *HGUI* still does not run *hypgen* and *hypgen.cray* properly on the remote machine.

A. Check the dimensions in *hypgen*. Large dimensions may have been declared in *hypgen* that requires huge run time memory which exceeds the memory limit for interactive jobs on the remote machine.

4. Concluding Remarks

HYPGEN and *HGUI* are still under constant development to improve the robustness of the algorithm, to include more types of grid topologies, to automate as many parameter inputs as possible and to enhance the user interface such that user effort is reduced. Comments and suggestions are most welcome.

For questions, comments, suggestions and bug reports on *HYPGEN* , please contact

```
William M. Chan
MCAT Institute
NASA-Ames Research Center
MS T047-2
Moffett Field, CA 94035
(415) 604-6607
E-mail address:  wchan@nas.nasa.gov
```

For questions, comments, suggestion and bug reports on *HGUI* , please contact

```
Ing-Tsau Chiu
Overset Methods Inc.
NASA-Ames Research Center
MS T047-2
Moffett Field, CA 94035
(415) 604-3857
E-mail address:  chiu@nas.nasa.gov
```

The source code, executable and documentation are provided with the *HGUI* package. If the user wishes to recompile *HGUI* , the "Panel Library" is needed. The "Panel Library" can be purchased from

```
COSMIC
The University of Georgia
382 East Broad Street
Athens, GA 30602
(404) 542-3265
FAX: (404) 542-4807
E-mail address:  service@cossack.cosmic.uga.edu
```

References

[1] Steger, J.L. and Rizk, Y.M., Generation of Three-Dimensional Body-Fitted Coordinates Using Hyperbolic Partial Differential Equations, NASA TM 86753, 1985.

[2] Walatka, P.P., Buning, P.G., Pierce, L. and Elson, P.A., PLOT3D User's Manual, NASA TM 101067, 1990.

[3] Chan, W.M. and Steger, J.L., Enhancements of a Three-Dimensional Hyperbolic Grid Generation Scheme, *Appl. Math. & Comp.* **51**, 181–205, 1992.

[4] Thompson, J.F., Warsi, Z.U.A. and Mastin, C.W., *Numerical Grid Generation, Foundations and Applications*, North-Holland, New York, p.306–309, 1985.

[5] Kinsey, D.W. and Barth, T.J., Description of a Hyperbolic Grid Generation Procedure for Arbitrary Two-Dimensional Bodies, AFWAL TM 84-191-FIMM, Wright Aeronautics Laboratory, Wright-Patterson AFB, Ohio, 1984.

[6] Kordulla, W. and Vinokur, M., Efficient Computation of Volume in Flow Predictions, *AIAA J.* **21**, 917–918, June, 1983.

[7] Buning, P.G., Chan, W.M., Renze, K.J., Sondak, D., Chiu, I.T. and Slotnick, J.P., OVERFLOW User's Manual, Version 1.6, NASA Ames Research Center, Moffett Field, CA, 1991.

[8] Vinokur, M., An Analysis of Finite-Difference and Finite-Volume Formulations of Conservation Laws, *J. Comp. Phys.* **81**, 1–52, 1989.

[9] Obayashi, S., Free-Stream Capturing in Fluid Conservation Law for Moving Coordinates in Three Dimensions, NASA Contractor Report 177572, 1991.

[10] Parks, S.J., Buning, P.G., Steger, J.L. and Chan, W.M., Collar Grids for Intersecting Geometric Components Within The Chimera Overlapped Grid Scheme, AIAA Paper 91-1587 in Proceedings of the AIAA 10th CFD Conference, Honolulu, Hawaii, 1991.

[11] Pearce, D.G., Stanley, S.A., Martin, F.W., Gomez, R.J., Le Beau, G.J, Buning, P.G., Chan, W.M., Chiu, I.T., Wulf, A. and Akdag, V., Development of a Large Scale Chimera Grid System for the Space Shuttle Launch Vehicle. AIAA Paper 93-0533, 1993.

[12] Tristram, D.A., Walatka, P.P. and Raible, E.L., Panel Library Programmer's Manual, NASA Report RNR-90-006, 1990.

Appendix A.

Chronology of *HYPGEN* Versions

New HYPGEN Features in Version 1.1

(1) The default name for the input surface grid is now "surf.i."

(2) IDISS has been renamed IMETH.

(3) Extra robustness at very sharp convex corners is provided by new algorithms (IMETH = 2 and 3).

(4) The code is now able to treat the degenerate case of a zero-thickness plate topology.

(5) The grid lines crossing check has been replaced by a Jacobian check using the same algorithm as that in the flow solver *OVERFLOW* . Negative Jacobians are reported.

(6) The coordinate that is constant in a 2-D grid case is now allowed to be non-zero. This is determined from the supplied "surface" grid which is a coplanar 2-D curve on an x, y or $z =$ constant plane.

(7) The need for post-process smoothing at corners is removed with introduction of a blanking function.

(8) The code tests for consecutive coincident points on the surface grid. If any such points are found, an error is reported and the program is terminated.

New HYPGEN Features in Version 1.2

(1) All parameter statements in the code have been removed except the one in the main program. Only the maximum number of points in the 3-D grid, a 2-D J-K plane and a single direction need to be specified (see §2.5).

(2) The output grid is now in *PLOT3D whole* format.

(3) Slight improvements have been made to the algorithm for IMETH = 3.

(4) Tridiagonal block solvers for both periodic and non-periodic cases have been rewritten. The new routines require less memory to run and the operation counts have been reduced by making use of the special structure of the block matrices that appear in *HYPGEN*.

(5) Error messages for grid quality checks are now in a more compact form and a summary table of the number of bad cells at each L-shell is provided.

(6) It is now possible to apply a free floating or constant plane boundary condition at a singular axis point boundary (i.e., all points at the boundary on the surface are coincident).

(7) The filename for variable far field distance input has been changed from *hyg3dzm.i* to *zetamx.i*. The code can now cope with a variable far field distance that is a function of both J and K. The user is warned that this option has not been fully tested. A known problem is that it will not work if a free floating edge boundary is present.

New HYPGEN Features in Version 1.3

(1) The scheme used to generate the points at $(J, K) = (1, 1)$, $(1, kmax)$, $(jmax, 1)$, $(jmax, kmax)$ have been improved to provide more robust results.

(2) The axis logic has been improved slightly by doing weighted averages.

(3) A small bug in subroutine angmet2 has been fixed (the bug shows up when neighboring grid points are extremely close together in the interior).

(4) The algorithm for the constant plane boundary condition has been enhanced such that the surface grid no longer needs to be orthogonal to the constant plane boundary. The resulting grid will follow the constant boundary plane near the boundary even if it is non-orthogonal to the $L = 1$ surface.

(5) The floating boundary conditions have been made more general as follows. Any combination of x, y or z coordinates can now be floated/fixed. Previous options allowed floating of all three (free floating) and floating of any two coordinates (e.g., fixed or constant plane in any one of x, y or z and extrapolate the remaining two coordinates). New options are now available at the boundary to fix any two of x, y or z and extrapolate the remaining one (see §2.3.1 for usage).

(6) The Jacobian computation in the grid quality checks has been modified such that symmetric values are obtained for symmetric grids (see §2.4.2).

(7) Problems associated with the *zetamx.i* file have been fixed.

(8) The problem in EPSIL for uniform spacing has been fixed.

(9) The data in the output file have changed slightly, e.g., always write out stretching ratios, restrict output of bad cells to max of 50.

New HYPGEN Features in Version 2.0

(1) The input parameter list has been simplified but the format is now incompatible with earlier versions of *HYPGEN*. The changes affect only the 12 parameters that were used to specify the boundary conditions. Now, only 4 parameters need to to be specified. The last line of the old input consisting of 3 axis parameters can now be omitted for cases with no axis boundary conditions.

(2) A variable far field distance, a variable initial grid spacing and a variable end grid spacing can be specified in the input file *zetavar.i*. The old variable far field input file *zetamx.i* is no longer used.

(3) A user-defined stretching function in the normal direction can be specified in the input file *zetastr.i*.

(4) More robustness is introduced for constant planes boundary conditions by a metric rotation scheme.

(5) Some new boundary conditions have been implemented: (a) floating collapsed edge with matching upper and lower sides for C and O-topologies, (b) constant interior planes in x, y or z. These new options have only gone through limited testing at this stage. Bug reports and comments on them are particularly welcome.

(6) The formats of all input files are automatically determined. The parameter IFORM is now used to specify the format of the output volume grid file only.

Appendix B.

Chronology of *HGUI* Versions

New HGUI Features in Version 1.1

(1) Correctly sets the boundary conditions for 2-D grid generation.
(2) Uses dynamic memory allocation.
(3) Allows basename to be optionally specified at startup.
(4) The grid generated by *HYPGEN* is converted from *PLOT3D* plane format to whole format.
(5) User's `.login`, in addition to `.cshrc`, is also checked for environment settings for the default window shell to launch *PLOT3D* .
(6) Uses only subsets of the generated grid for tetrahedron decomposition cell volume check to avoid unnecessary calculations when bad cells are found.
(7) Outputs proper *PLOT3D* script file for 2-D and 3-D grids.
(8) Expanded instructions on setting up *HGUI* and *HYPGEN* to run on different machines.
(9) Added 2-D examples.
(10) Fixed a bug in reading formatted grid files.

New HGUI Features in Version 1.2

(1) Added support for variable zeta max file.
(2) Fixed a bug in determining periodic BC in J.
(3) Two versions of executable are provided since IRIX 4.0 binary is not backward compatible with IRIX 3.3.
(4) Added check on consecutive coincident points.

New HGUI Features in Version 2.0

(1) The graphical user interface has been renamed from *UI* to *HGUI* . All occurrences of *ui* in shell scripts directories have been modified to *hgui*, e.g., *.uirc* is renamed *.hguirc*. The shell scripts *hypgen.4d* and *hypgen.ymp* have been renamed to *hypgen.sgi* and *hypgen.cray*, respectively.
(2) The old *"Boundary Conditions"* panel is replaced by a new one that accepts boundary condition types. Selection can be made from a pop-up menu or by typein.
(3) Under the main menu, (a) the input submenu is removed since it is no longer needed, (b) a new plottd submenu is installed to provide options to open a new *PLOT3D* shell and to do rehashing, (c) the *"Save"* button has been simplified to only ask for a filename to save the input parameters.
(4) The *"Grid"* and *"L-regions"* panels have been modified to allow users to (a) read in *zetastr.i* for a user-defined stretching, (b) select a variable far field, initial/end grid spacing, (c) specify more than 4 L-regions.
(5) The default location for the *PLOT3D* window is now preset. The default subsets for viewing the grid have been modified to be the surface grid together with the $j = 1$, $j = jmax$, $k = 1$, $k = kmax$ planes.